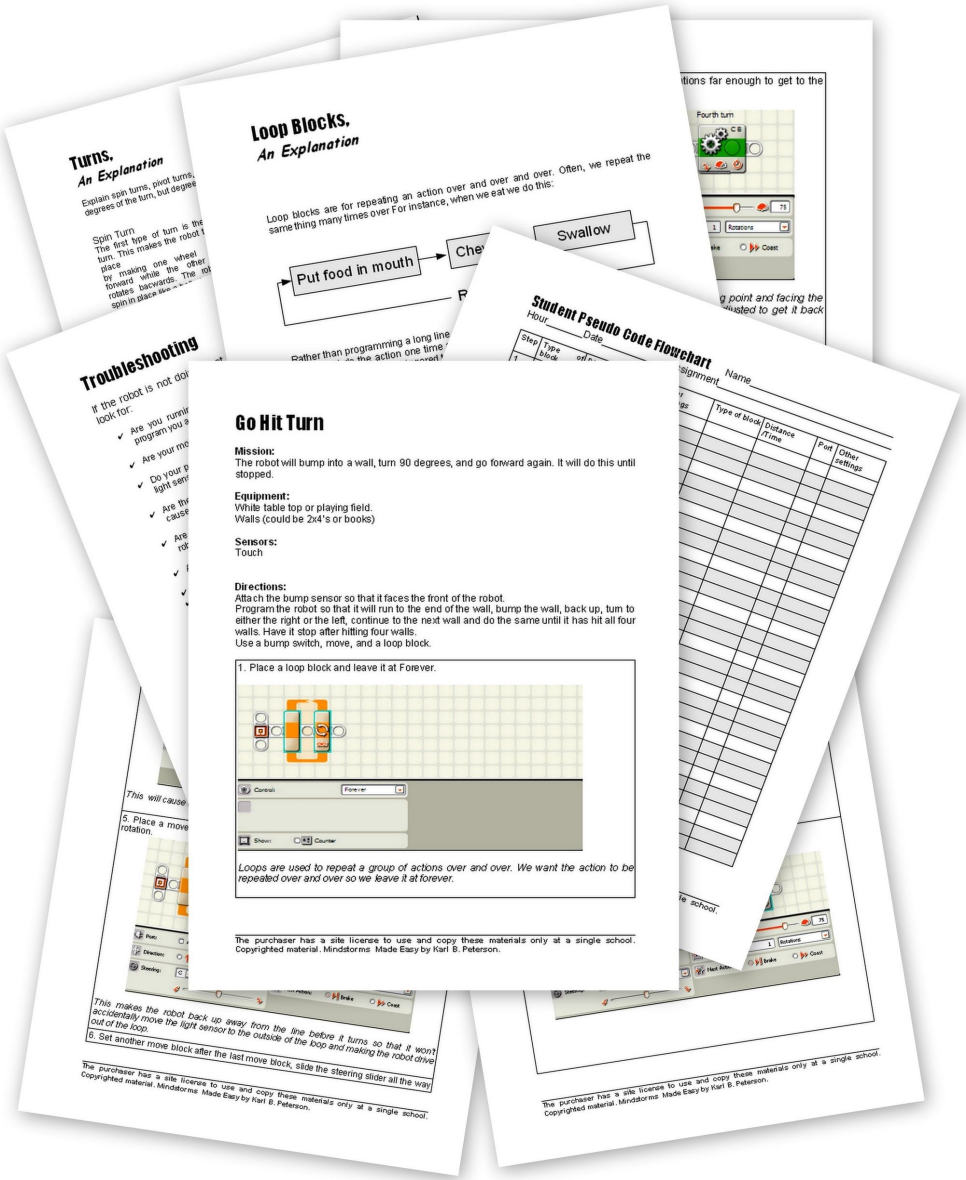
MINDSTORMS®

Made Easy

Lessons to teach programing in NXT-G®

by Karl B. Peterson



To my dear and wonderful wife and family who inspire me
and to my wonderful students who inspired this book.

MINDSTORMS®

Made Easy

Lessons to teach programing in NXT-G®

by Karl B. Peterson

Table of Contents

MINDSTORMS®

Made Easy

Lessons to teach programing in NXT-G®

by Karl B. Peterson

The goals of this book:

- Start with easy lessons true beginners can do and have a sense of accomplishment.
- Use easy lessons that can be completed quickly.
- Lessons build slowly so all the students can learn how to program.
- Prepare students to program for the FIRST Lego League®.
- Do it inexpensively.
- Easy setup and a variety of materials so teachers can teach NXT-G programming without a lot of preparation because teaching and spare time don't usually go together.
- Make it fun.

Introduction

Maybe you are facing a challenge that I faced a few years ago. I was assigned to teach a class in robotics. I was excited but panicked since I have always liked science and wanted to learn about robotics, but I had never done any robotics in my life. I was a former English and drama teacher, for heaven's sake! Most of the time in teaching school we teachers can think back about how we were taught the subject when we were in school. Robotics is a new enough subject that most of us never studied it in middle school or high school.

As I started looking for lesson plans to teach the classes (and me) about robotics, I found that there was very little out there that did not cost a bundle. Also, many of the lessons I could find assumed the class and I already knew how to program the robots. They seemed to be written by scientists for scientists and the authors assumed that my students and I were a lot more educated that we were about robotics. Also, the books I bought were more about *building* cool robots than learning how to *program* them. They also were not geared to a classroom setting. Not only that, but I was a teacher trying to figure out how I was going to come up with a grade to give these students. The books I found said nothing about this.

I knew students didn't have a long enough attention span to spend a week or more on a single assignment. I started out with four robots for a class of twenty-six students. I needed assignments that the students could do in about a half an hour to an hour in most cases and turn over the computer and the robot to the next student.

Also, I got a grant to start a First Lego League and I needed to find a way to teach my team how to program the robots to move, knock over, bump, or carry away various objects, so theses are the skills I wanted to develop.

Since I could not find a book or program to teach how to program Lego Mindstorms robots, I decided to write one!

This book of lesson plans is my solution to my dilemma of trying to find a way to teach Lego Mindstorms NXT programming to classes of middle school students. This book will give you a relatively painless way to learn the programming you and your class need to program enough to compete in the First Lego League competitions as well.

Lego Mindstorms have swept through elementary and middle schools as a way of giving hands on experience for science, technology, engineering, and math (STEM), but how to do this? I hear a lot of teachers say, "I just hand it over to the students and let them figure it out." While this may be one method of survival in the classroom, it is hardly the best. I hope this book will not only help you teach your students, but also teach yourself along with them! You will benefit from the many hours I spent figuring out a way to make robots accessible to the regular middle school student sitting on the back row. Using this book will help you teach your students a great deal about programming and still have time to play the Wii with your family.

This program is made for the busy teacher who wants to incorporate Lego Mindstorms in the curriculum. It has many assignments that can be done by a variety of levels of students. The assignments are meant to slowly build on each other and revisit various programming basics so the students can master the skills necessary to program the Lego Mindstorms robots.

Purpose

To equip teachers to teach NXT-G programming to middle school students as easily and painlessly as possible.

Set Up

Needed Equipment

Mindstorms NXT kit for each group of students (1-5 students in each group--the fewer the students per computer the better).

A computer for each group.

A tabletop set up for testing the robots consisting of a four foot by four foot sheet of plywood and two by four sides on it.

10 two by four pieces eleven and a half inches long.

The lessons build around a simple to construct design of a robot that takes very few parts. As you need to add sensors, you will need to add parts to connect them to the robots.

Teams

Each individual student needs to write his or her own program. The class will be divided into teams of about 4 or 5 students each. Each team should have at least one robot and one computer to use. Though it is ideal to have one robot for each student, two robots for each computer can be used by letting one student work on a program while the other student is at the testing table seeing what needs to be done to improve the program. That way, each student can be working without getting in the way of the other. The purpose of the team is to allow the students to work out some of their problems before taking them to the teacher. This allows the teacher more time to monitor the class as a whole instead of spending large amounts of time with one struggling student while the rest of the class gets off task. It is a great idea to offer extra credit or some other reward for each team whose members have all completed the assignment. That way, they will help the student or students who are struggling. My rule is that a student can help talk through the steps to do the assignment as long as the struggling student is the one using the keyboard and mouse. Don't let other students do the work for the struggling student.

Keeping track of grading

Many schools recommend the students bring their agendas to every class and use them so they write out the assignment name on the day they do it. I initial the assignment in their agendas. This gives them a written “receipt” of them doing the assignment.

Organizing Robots

Keep as much of the kits locked up as you can. I start class with only a simple robot setup for the students to use and I don't even have any sensors hooked on to it at the beginning. Only hand out the various parts as needed to do the exercise. That way, fewer parts will “walk away.” Lego parts are as valuable as gold to middle school students and some students will decide to use the school's Lego parts to supplement their own collections. Many of the teachers I talk to mention many of the Lego parts disappearing so I limit the student's access to them. Several weeks into the class, I open up a box that has the sensors in it and enough parts and cords that the students can hook up whatever they need without so many parts available that the few dishonest students can't take too much. The students love building with Legos and I think there is real value in doing it, but I chose to make my class more about programming than building. If a student can learn to program really well, they can become good builders later.

This is a picture of the sensor box I have available to the students. They can get the sensors they need, a cord to plug them into the intelligent brick, and some pieces to attach the sensors to the robot.

Use round stickers of various colors to mark the sensors, motors, and intelligent bricks. This allows you and your students to know which robots are being used and by whom. Also, since First Lego League does not allow you to have outward markings on your robot, you an remove the stickers when you go to a First Lego League competition. The stickers are found in the office supply section of most big box stores. They are by the price stickers used for garage sales.

As a reminder, these lessons are not meant to teach the building of Mindstorms robots; it is to teach the programming of the robots. I used the robot design found at NXTPrograms.com. Dave Parker has done a great job in showing a number of very creative robot designs. The one used in these materials is called the 5 minute Robot and is found at: http://www.nxtprograms.com/five_minute_bot/steps.html. It is a simple design with only a few parts. This lends itself to students to make changes as necessary and it is easy to repair in case the robot comes apart.

Troubleshooting

If the robot is not doing what you want it to do, there are several things to look for:

- ✓ Are you running the right program? Be sure the program on the brick is the program you are working on.
- ✓ Are your motors and sensors plugged into the right port?
- ✓ Do your program blocks always linked to the same port? For instance, does your light sensor look at port 1 one time and port 2 the next?
- ✓ Are the wheels rubbing against the motor or another part of the robot? This will cause the wheels not to turn or turn slowly.
- ✓ Are any cables hitting into any of the swiveling wheels in the back and causing the robot to turn when it should be going straight.
- ✓ Are the cables completely plugged in? If one motor cable is not plugged in, the robot will move a tiny mount and then quit.
- ✓
- ✓ Are the batteries charged? When the batteries are getting low, the robot will start to act very strangely.
- ✓ Sometimes you did everything right and the program will not work. You will need to remove the part of the program that does not work and redo it. Other times, computer program will not run right on the robot and you may need to close the program and rewrite it.
- ✓ If the program says the robot is not attached and it is, try unplugging the USB cable and plugging it back in. Check that the cable is also plugged in completely. Also, make sure the robot is turned on. If you have tried these things and they did not work, try a different USB cable. They can get damaged and not work right anymore so they may need to be replaced.
- ✓ Use the view feature to see if the sensor is working right.

Practice Table

The practice table is where you run the programs after the robot has been programmed. Don't run the robots on a table top where they can fall off or on the floor since they can get stepped on. If you have a First Lego League table, you can use that. Just cut a two by four about forty-five inches long and place down the middle so that the table is now divided into two four foot by four foot sections. If you don't have a practice table, you can take a sheet of half inch plywood and four two by fours and build your own practice table. You can make one large practice table that will give you two areas to practice or you can have the plywood cut into two four by four foot sections that will be one practice area each. This is easier if you plan on taking your table down and setting it up over and over or if your storage area is not that large. You may want two or more tables so the students will not need to be waiting for each other too long to do the missions.

Materials:

4 - 2x4x8 boards
1sheet of ½ inch plywood
1 box 3 inch course drywall screws
1 quart white satin latex interior paint.
8 feet of heavy thickness, clear plastic (found in sewing stores or in the sewing department of Walmart. It is used to protect furniture. It comes on long rolls.

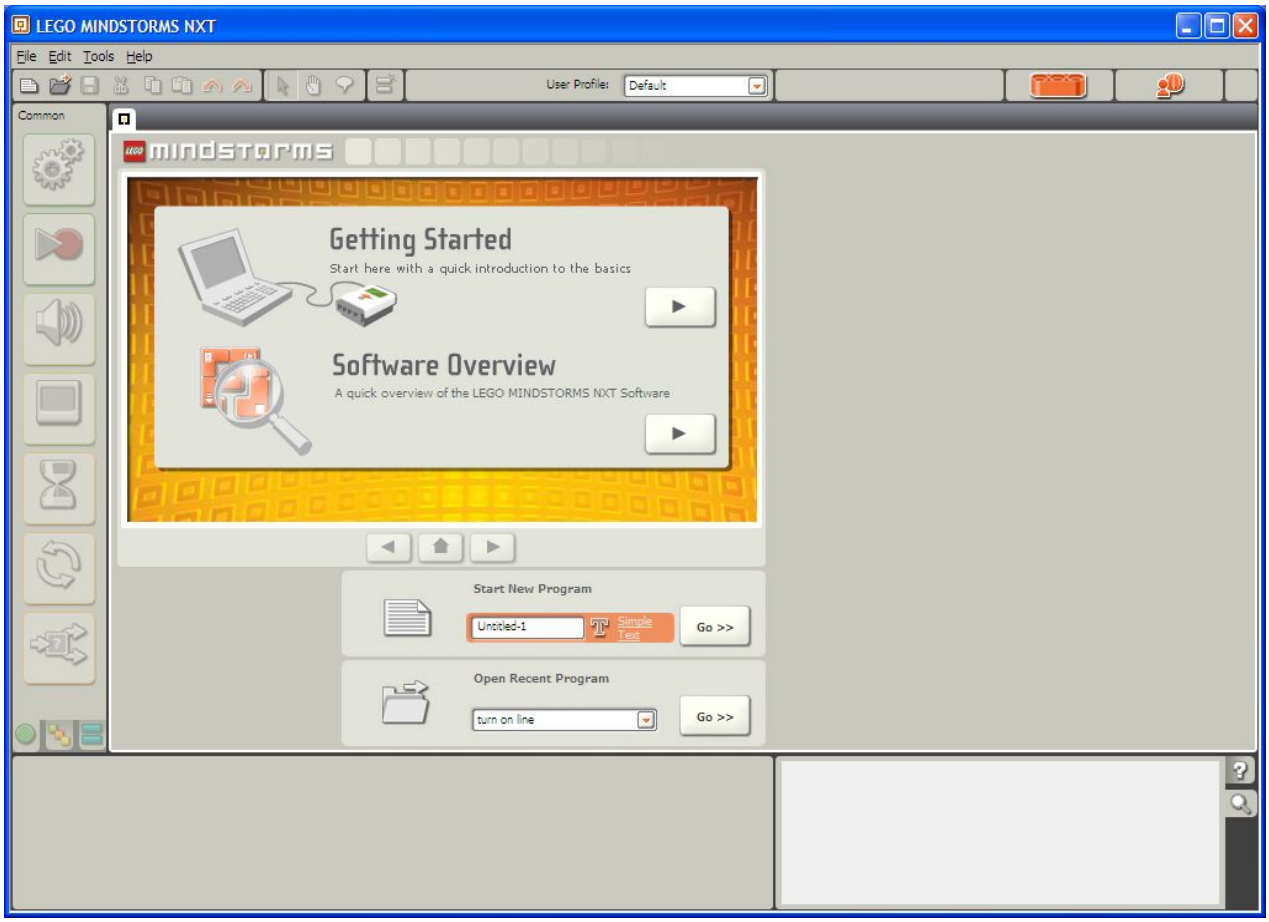
Equipment: brushes, electric drill, 1/8 inch drill bit, Phillips head drill bit.

Steps:

1. Cut 2 of the 2x4's into 45 inch lengths. You need three 45 inch pieces.
2. Lay the plywood down on the floor with the smoothest side down.
3. Place the 8 foot long 2x4's down either side of plywood long ways.
4. Place 2 of the 45 inch pieces of 2x4's width ways on the ends of the plywood.
5. Screw through the plywood and into the 2x4's about every foot or so. Stay back from the ends of the wood about 3 inches to not split the wood.
6. Pre-drill holes through the 8 foot long boards and into the 45 inch boards.
7. Insert screws into the holes.
8. Flip over the table.
9. Measure the middle of the board length-wise so you divide the table into roughly 4x4 foot sections, Mark it.
10. Place a 45 inch piece of 2x4 between the 2x4's, pre-drill holes 2 holes in each side and insert screws.
11. Paint it all white.
12. Let it dry
13. Place it on saw horses or on a table.
14. Cut the plastic to fit.

GETTING STARTED

When you first load the disk that came with the robot and you start the program, you will first see this first screen.

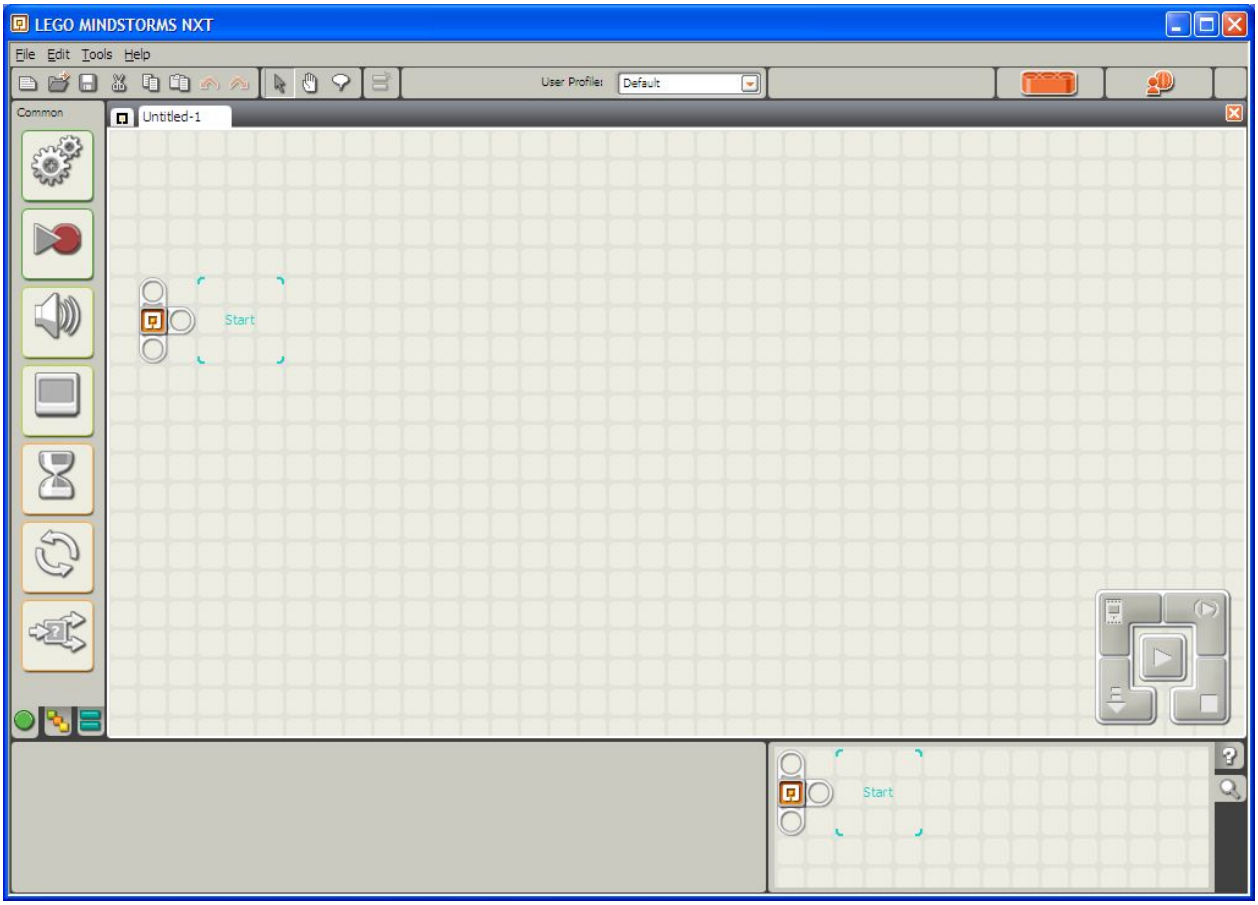


The Getting Started and Software Overview are good introductions to Lego Mindstorms. Go through them if you are not already familiar with the program.

Lower on the screen and in the middle is the Start New Program section. You can leave the new program named Untitled-1, but you will do better to use a more descriptive title. I like to use the name of the assignment or mission, your name, and the date. Something like **2.Loops.Peterson.6.10.** That way, you can easily find your program again. It's also a good idea if you are writing a particularly long program, that you save it often and rename it so if you make changes to the program that make it worse instead of better, you will have your old work to go back and use again.

Below the Start new Program is the Recent Program box. It will contain the last several programs you have done so you can use them easily without needing to do a lot of

searching. Once you either start a new program or open another one, you will be taken to a new screen that will look like this:



You see near the upper left you have File where you can open, save files.

If you went through the Getting Started and Software Overview introductions, you will know how a little to get you started. Don't worry about remembering every bit of it. This book will walk you through the various programs every step of the way.

If your opening screen does not look like this, be sure to click on the little green circle on the lower left side. It will take you back to this screen.

The next part is this box.

The button that is most important is the lower left one. This downloads the program from the computer to the robot using the cord provided in the kit. The robot must be on or it will not work. You will then need to run the program after it is downloaded. It is a good idea NOT to use the middle play button since it will download the program and immediately start running the program. This can cause your robot to drive right off the table and fall to the floor. Use the lower left button.

This brings us to another really important rule:

Run the robot only on the practice table.

The practice table has sides that will keep the robot from rolling off the table. If you are at home, you may be willing to use the robot on the floor, but you always take the chance of someone kicking it by accident or on purpose. The robots are expensive; take good care of them.

Programming

The robot can't do anything unless you tell it to. How do you do that? You give it a set of instructions called a program. The robot follows the program and does what it is told to do. Programming is really two different tasks. The first task is the most obvious. The programmer has to learn the programming language to tell the robot what to do. You have to use the language the robot understands. Just like if you were to try to give directions to someone who does not speak your language, it would not matter how good your instructions were, the person would never get to where you were explaining to get to. The robot is the same way. You have to speak its language. This means you need to learn how to use NXT-G which is the programming language that comes with the Mindstorms NXT and is the language the First Lego League requires the contestants to use. The nice thing is that the NXT-G is a pretty easy language to learn and use. You don't need to learn special word order and punctuation to make it work. It uses a variety of blocks that are lined up on a bar. You then adjust the settings of the block to tell the robot what to do.

The problem is the robot only does what you *tell* to do—not what you *want* it to do. You not only need to learn the language of the robot, you need to be able to take the parts of NXT-G and put it together to tell the robot what to do. There is an old computer programmer saying that says “garbage in—garbage out.” It means that if you give it bad code or bad information, the computer will give you garbage back. A good programmer is able to take a difficult task and break it down into a sequence of small actions and decisions that will get the task done. Long before you start programming, you need to decide what you want the robot to do and break down this big task into a bunch of little steps. Each step gives the robot one little thing to do so that it will accomplish what you want it to accomplish. You start by making a flowchart of what you want it to do. You have mental flowcharts for everything you do. For instance, you know the steps it takes to get dressed everyday and you know that if you put your shoes on before you put on your pants, you will have a pretty tough time getting dressed. Part of your flowchart might look like this:

Part of the Flowchart to Get Dressed in the Morning.	
Step	Task
1	Put on pants
2	Put on socks
3	Put on shoes

You have do do things in a certain order to make it work. That is what a flowchart is all about. Builders need flowcharts to build a house. The painters can't paint the walls until the walls are there to paint. You can't put on the roof if there is no walls to hold up the

roof. You need to follow a sequence to make it all work. Now notice in the Task column that there isn't any special code. It is just written in ordinary English. We call it pseudo code.* “Pseudo” means false. In other words, the flowchart is in “false code.” A programmer writes in plain English, or pseudo code, to work out the steps that need to happen in a program in an easy to understand way so that changes can be made without having to read through line after line of hard to understand computer code. It is just an outline or a rough draft of the program.

* Note: “Pseudo” is pronounced “SUE-do.” The “p” is silent. The “sue” is pronounced like the girl's name Sue and “do” like bread dough.

It will save you a lot of work if you first think about what exactly you want the robot to do and break it down into all the little steps that the robot will need to do in order to complete the task.

One thing to remember with programming is that there are many ways to accomplish a task. One person will do it differently than another. Remember that as long as it works, you have done what you needed to do. With that said, it is a good idea to try to keep it as simple as you can. The saying “Keep it simple, silly,” is good advice.

Use the Pseudo Code Flowchart on the next page to write out the steps of what needs to happen in your program. Use the separate columns to show what happens in each part of a Switch. When you first write a program, use only the non-grayed rows to write your program. Use the grayed rows to add steps if you find you need to add steps to improve your program.

Student Pseudo Code Flowchart

Name_____

Hour_____Date_____Assignment_____

Step	Type of block	Distance /Time	Port	Other Settings	Type of block	Distance /Time	Port	Other settings
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								

Go

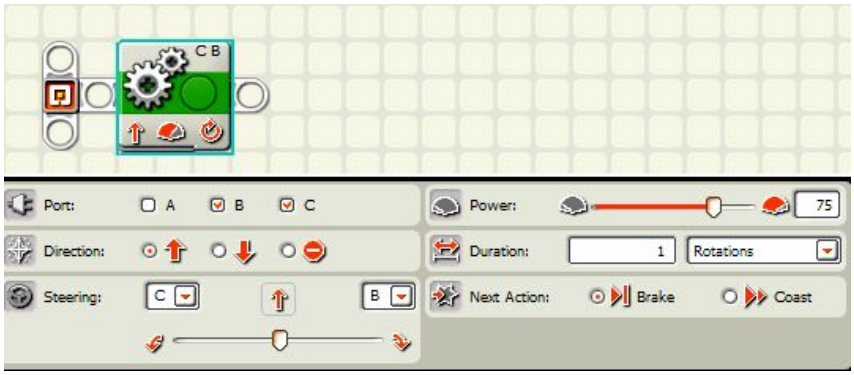
Mission: The robot will travel the distance assigned and stop.

Equipment:
White table top or playing field.
12 inch ruler

Sensors:
none

Notice at the top of the move block the letters C and B. This means that the control block will control motors plugged into ports C and B. You can leave it like this or you can click on the radio buttons in the middle left side of the illustration where it says "Port."

"Direction" lets you decide to have the move block move the motors forward, backwards, or make them stop.



Rotations lets you set how many times the wheels will rotate. You can use decimals to have parts of a turn.

One the bottom is the steering slider. You can slide this one way or the other to make the robot turn.

On the lower right you can see brake or coast. Brake will make the wheels stop moving right after it finishes the number of rotations it is told to do. Coast will let the wheels coast to a stop after it finishes the turns. It is less exact.

Go

For the teacher

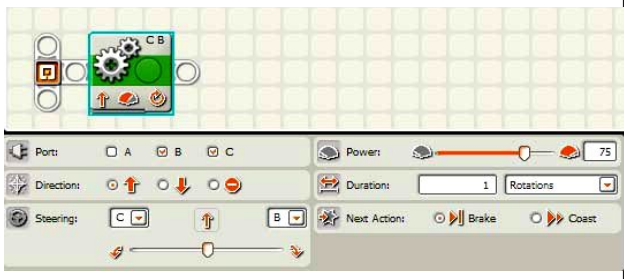
Directions:

1. The teacher assigns each student a length for the robot to travel. Each student in a group should have a different length to achieve.
2. Have each student take turns programming the robot to travel the assigned length. Some experimentation will be required.
3. When the student has achieved the distance, give or take a half an inch, go to the teacher to witness the successful completion of the assignment.
4. Present to the teacher the student's engineering journal of the distance and the number of rotations it took for the robot to move that distance.
5. The teacher initializes the student's grading paper and marks it in the grade book.
6. The students repeat each step until all the students in the group have accomplished the goal.

Here are some measurements for the students to use.

Length to travel	Wheel rotations
12 inches	
7 inches	
5 inches	
10.5 inches	
9 inches	
8 inches	
6.5 inches	
11 inches	

An explanation about the directions . . .

On the top of the box to the right, you will see the step number and the instructions of what you are to do.	1. Place a move block at the start of the program bar and set it for the number of rotations to go the assigned distance.
After that, you see what the program and the dialogue box under it should look like after you finish making any changes you were told to make	
At the bottom of the box, you see in italics an explanation of what the directions told you to do and what it means. This helps you to understand what you are programming the robot to do.	*Notice that the arrow pointing up has an orange dot by it. This means that the robot will move forward.*

Go and Back

Mission: The robot will travel the distance assigned and then move in reverse to the original starting point.

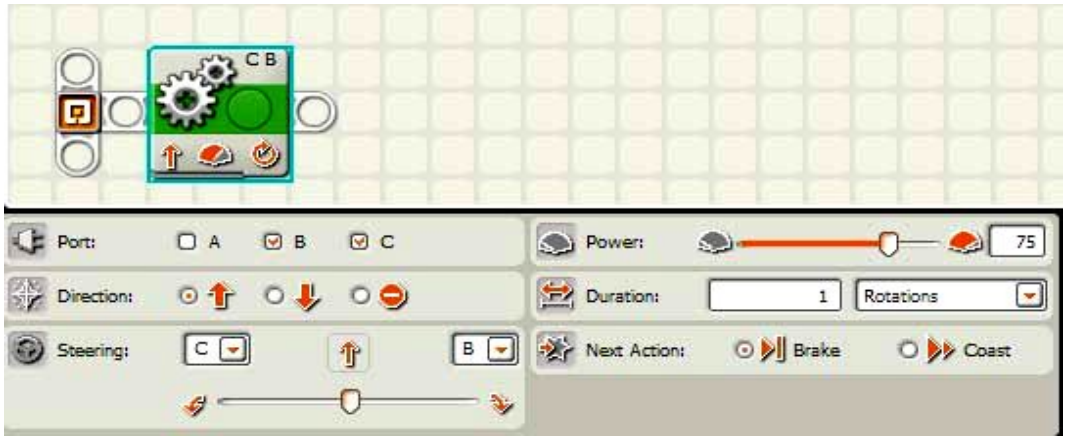
Equipment:
12 inch ruler

Sensors:
none

Goals:
Accomplish the mission
Record in the engineering journal the distance and the needed rotations to achieve it.

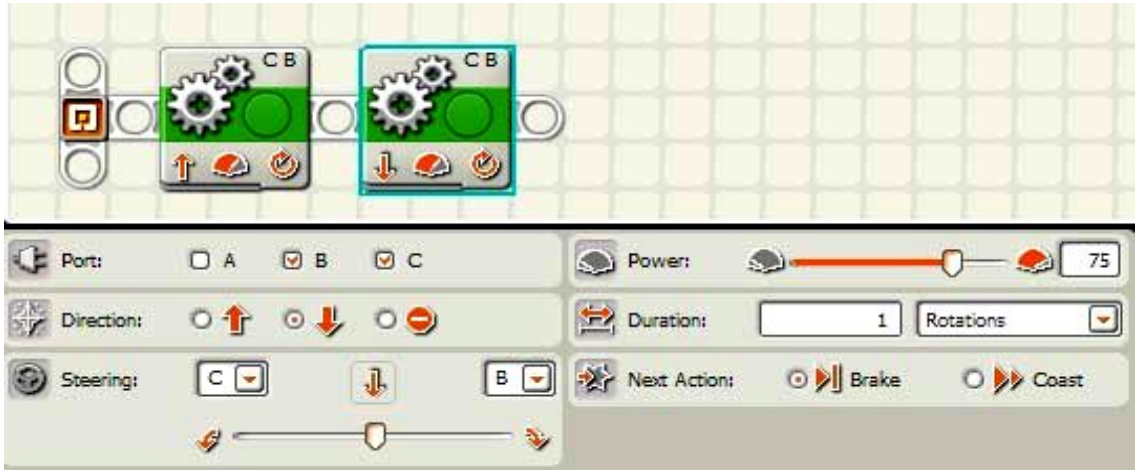
Directions:

1. Place a move block at the start of the program bar and set it for the number of rotations to go the assigned distance.



Notice that the arrow pointing up has an orange dot by it. This means that the robot will move forward.

2. Place another Move block after the first one and set it to go in reverse the same distance as the first one.



This time you clicked the arrow pointing down. This means the robot will move backward.

Secret to success: Figure out how many rotations it takes to go one foot and write it down. This will help you in future assignments so you can measure distances in later missions and then figure out how many rotations for the robot to move.

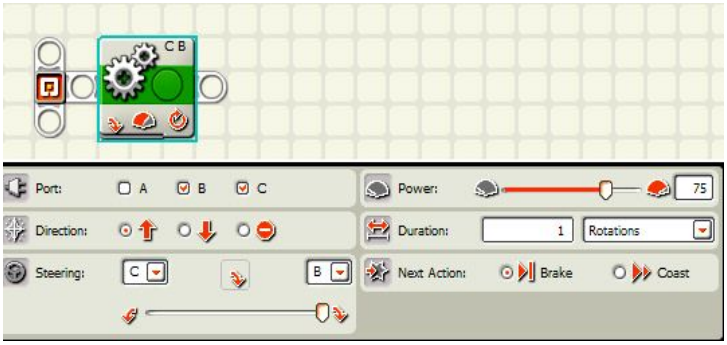
Turns,

An Explanation

Explain spin turns, pivot turns, and arcs (not very reliable). Explain that degrees are not degrees of the turn, but degrees of the rotation of the wheel.

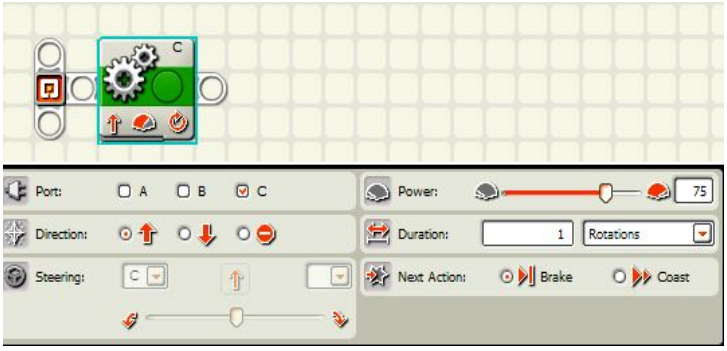
Spin Turn

The first type of turn is the spin turn. This makes the robot turn in place
by making one wheel rotate forward while the other wheel rotates backwards. The robot will spin in place like a ballerina.

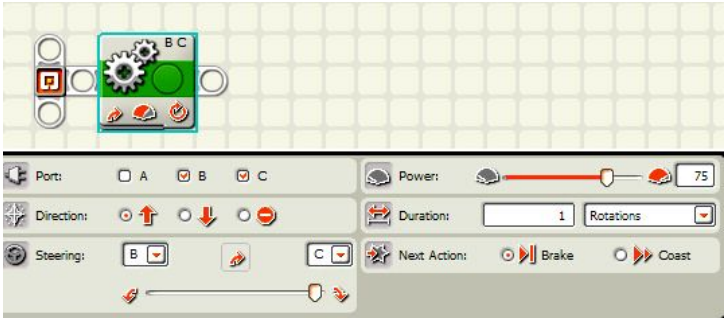


Pivot turn

The other type of turn is a pivot turn. This is where one wheel turns and the other one does not. It is like standing up and walking forward using the same foot and keeping your other foot on on the floor. The robot will turn while keeping one wheel in place. This pivot turn is done by only moving one wheel. See how only port C is picked?



Another way to make a pivot turn is to slide the steering slider in the lower left corner of the illustration almost all the way to the side so you can see just a tiny amount of the bar to the side of the slider.



Note: Sometimes students see that there are Degrees on the Rotations drop down box. These are degrees of the wheel turning. It is not degrees or turning the robot. So when you program the wheel to turn 90 degrees, it will turn 1/4th of a turn. 90 degrees will not turn the robot a ¼ of a turn. 90 degrees will turn the *wheel* ¼ of a turn. Degrees turn the wheel, not the whole robot.

Go Turn Back

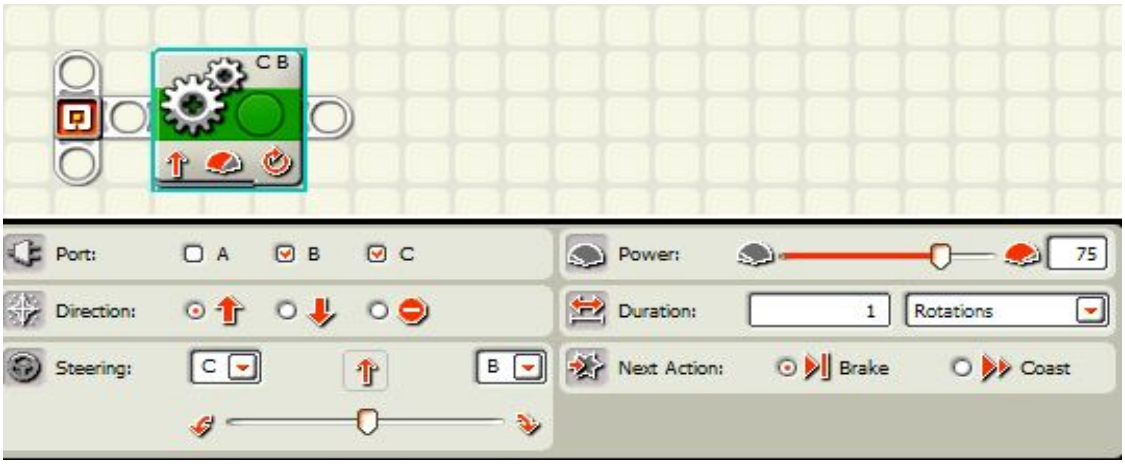
Mission: The robot will travel the distance assigned, turn 180 degrees to face the opposite direction and then travel to the original starting point.

Equipment:
White table top or playing field.
12 inch ruler

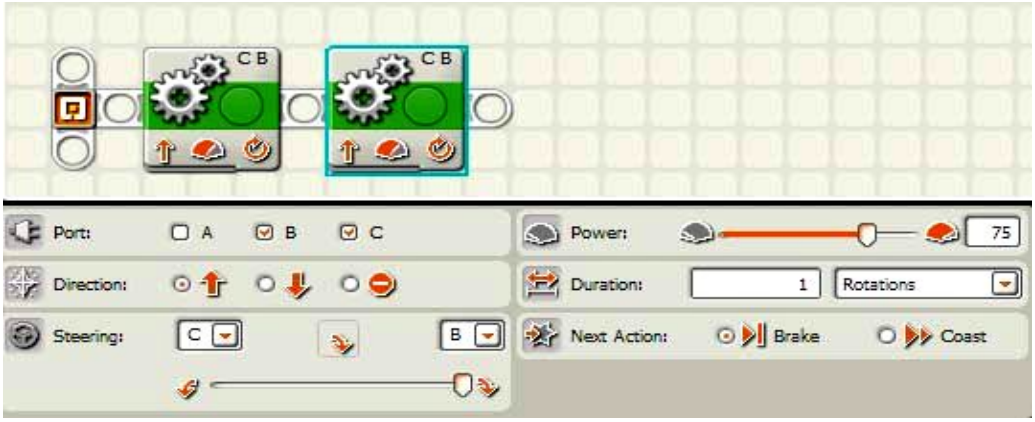
Sensors:
none

Directions:

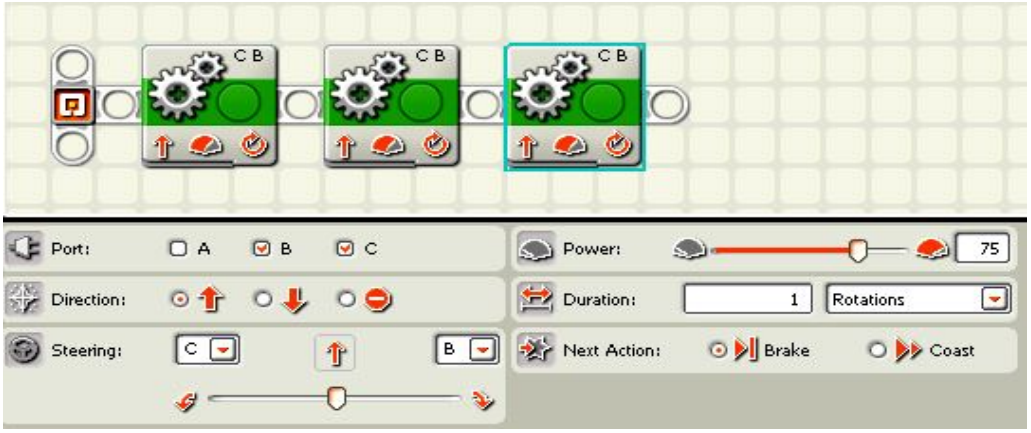
1. Place a move block at the beginning of the program bar and set the rotations for the distance you want it to go.



2. Place another move block and slide the steering slider at the bottom left all the way to one side to make the robot do a spin turn 180 degrees until it is facing the opposite direction. Experiment or use your engineering journal to set the number of rotations at the right amount to get it to do the turn correctly.

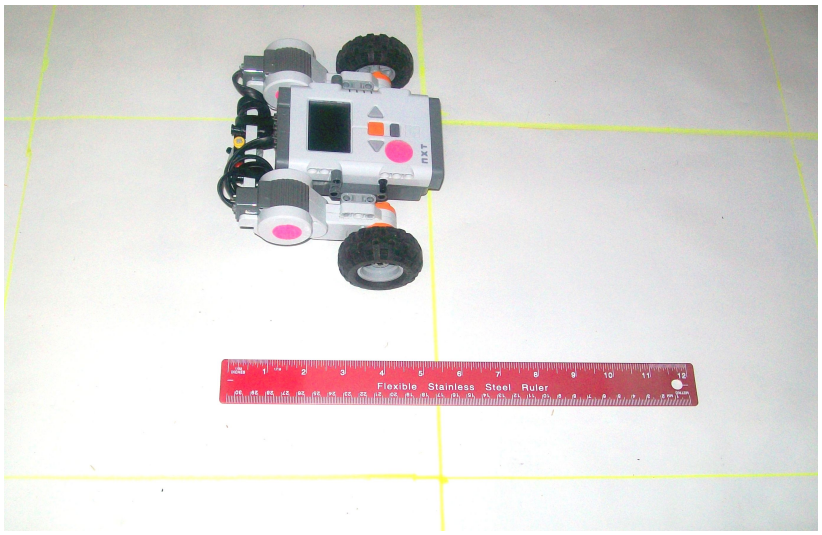


3. Place another move block and leave it going forward and set it to the number of rotations you set the first move block. If you got the rotations for the turn correctly and if you set the first and the last blocks to the same rotations, the robot will return to the same spot it started.



Secret to success: write down the amount of rotations it took to make the 90 degree turn. Keeping this information for future use will speed up the writing of other programs in the future. You won't have to take time figuring out how to make a turn on future assignments that way.

Circle the Ruler



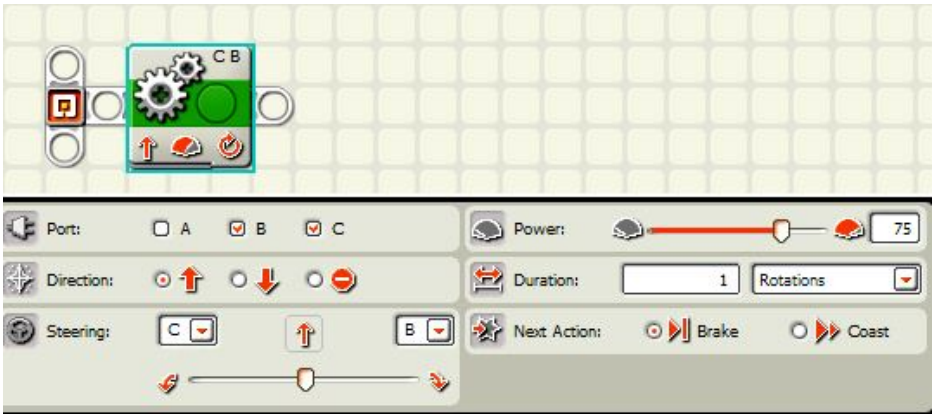
Mission: the robot will move around the ruler without touching it by moving in a large square or rectangle.

Equipment:
ruler

Sensors:
none

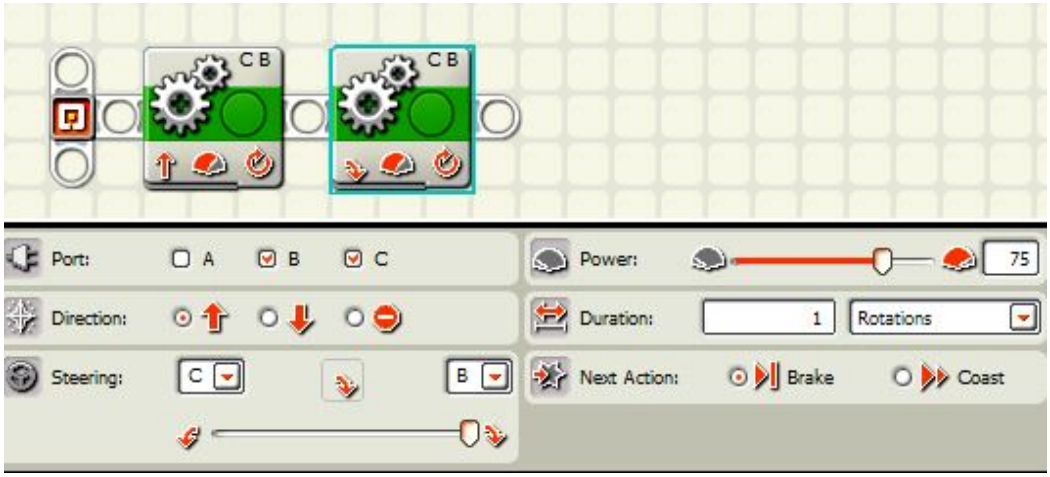
Directions:

1. Place a move block at the beginning of the program bar and set it to the distance you need to get past the ruler with some room to spare.



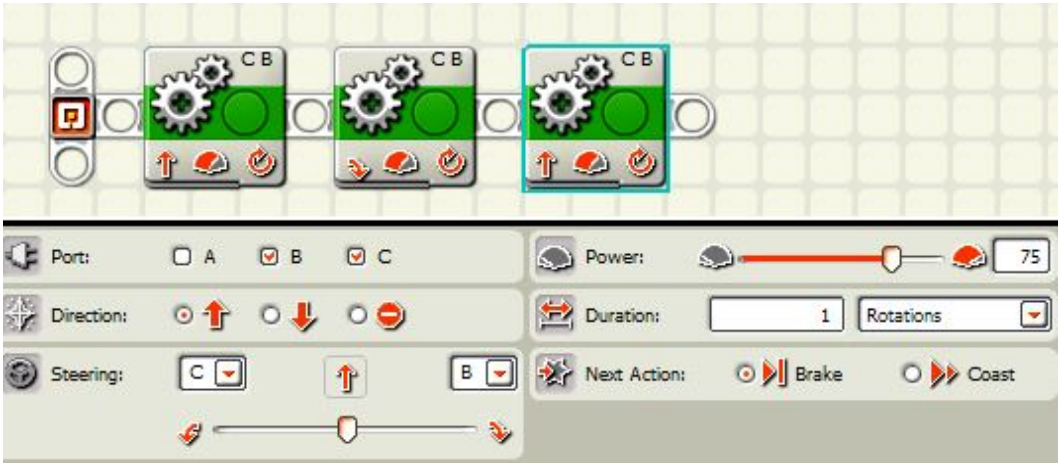
The robot travels in a straight line that gets the robot from one end of the ruler to the other.

2. Place another move block and slide the steering bar all the way to the side to make the robot turn in place.



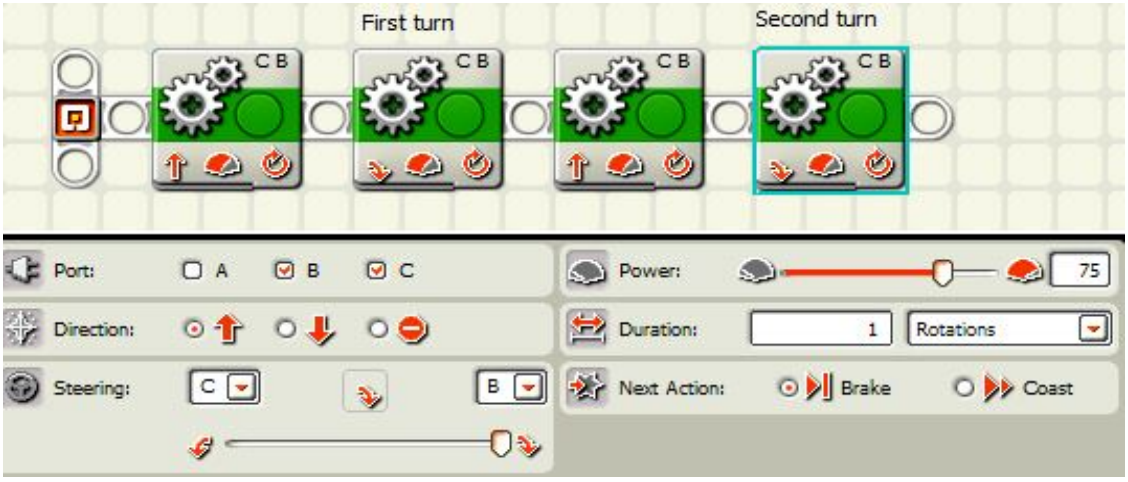
Sliding the slider all the way to the side makes the robot turn in place by spinning one motor forwards and one motor backwards.

3. Place another move block on the line and set the rotations far enough to get to the other side of the ruler with room to spare.



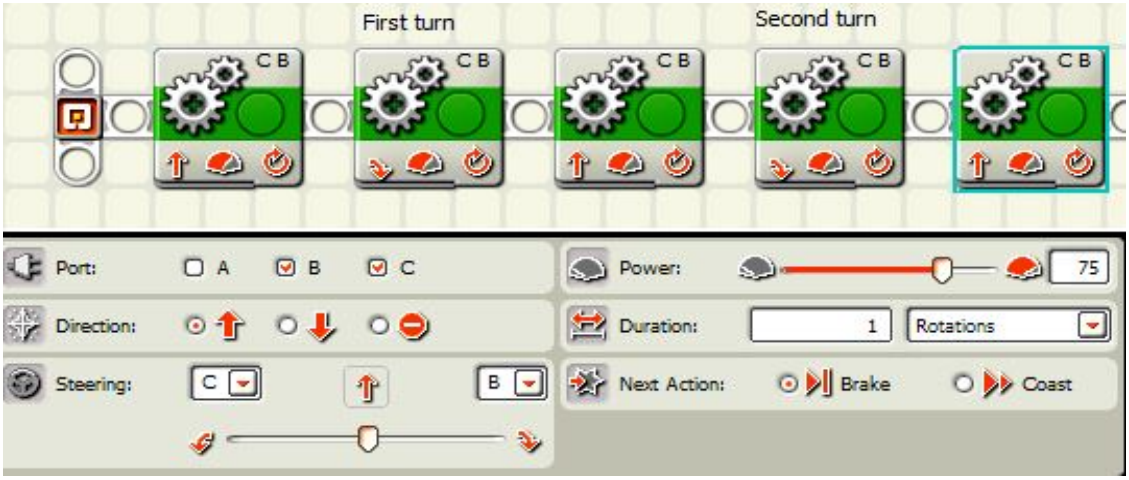
The ruler will now get to the other side of the ruler.

4. Place another move block and slide the steering bar all the way to the side to make the robot turn in place. This is the second turn.



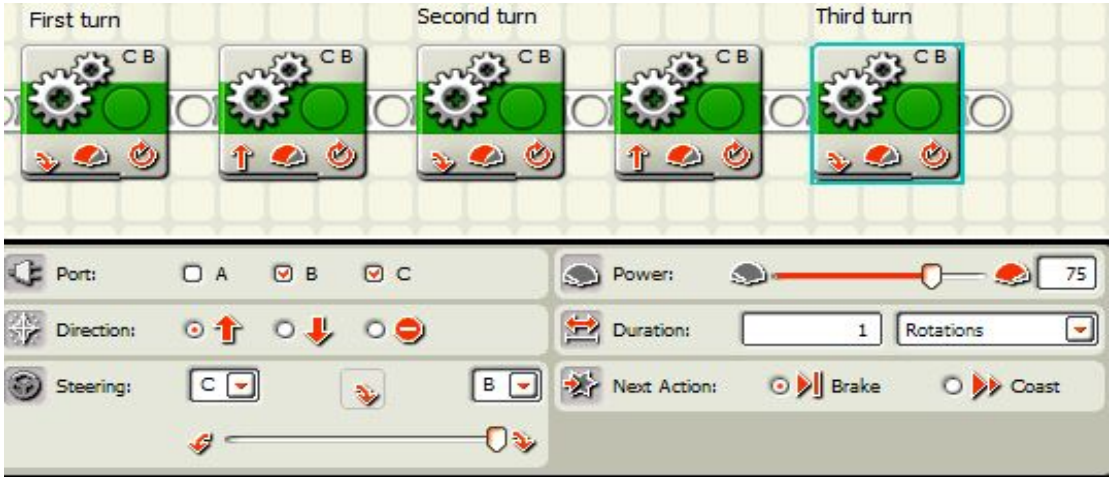
This turn makes the robot face the opposite way than it started.

5. Place another move block on the line and set the rotations far enough to get to the other side of the ruler with room to spare.



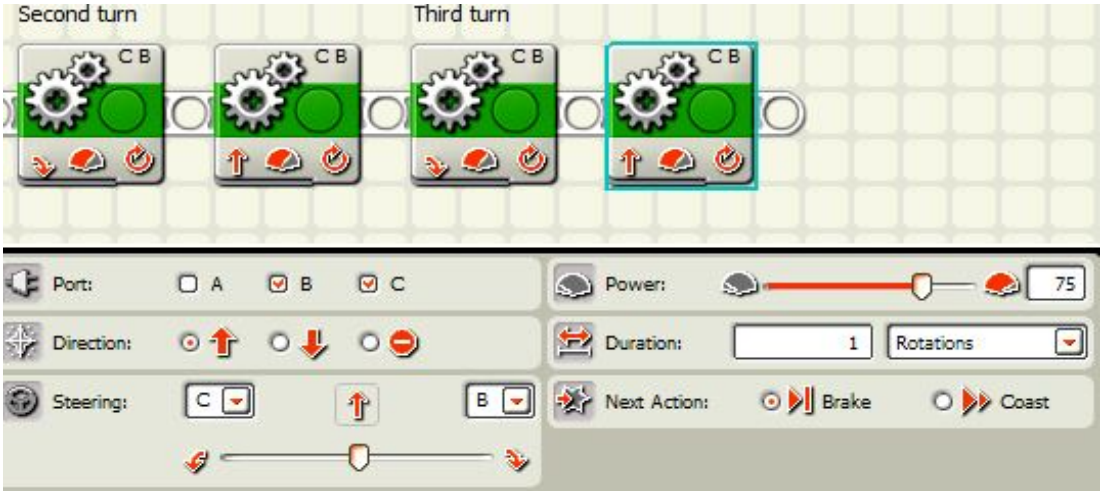
The robot will have finished a U-shape now.

6. Place another move block on the line and set the rotations far enough to get to the other side of the ruler with room to spare. This is the third turn.

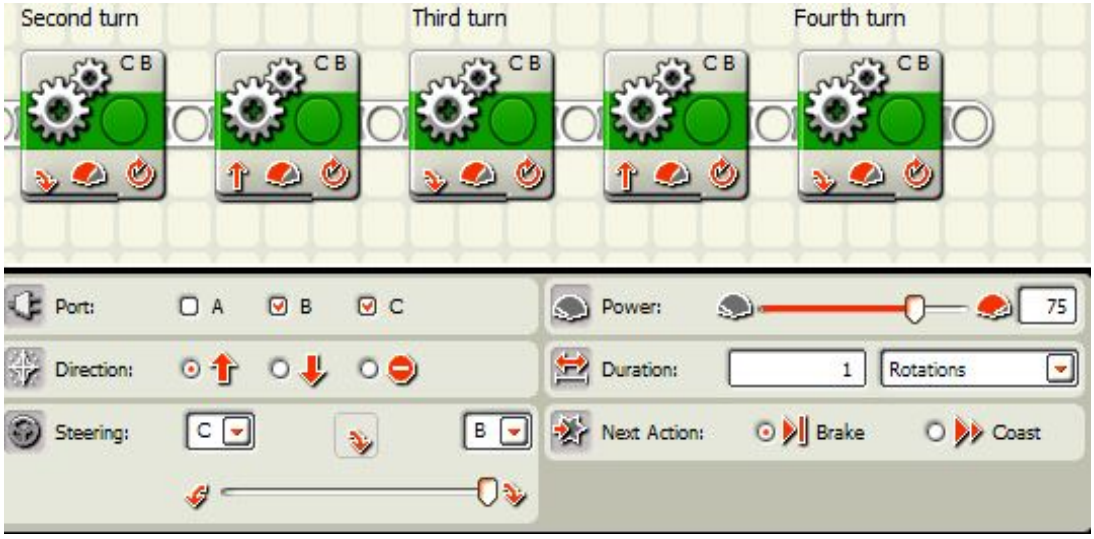


The robot will turn back towards the starting point.

7. Place another move block on the line and set the rotations far enough to get to the other side of the ruler with room to spare and get back to the starting point.

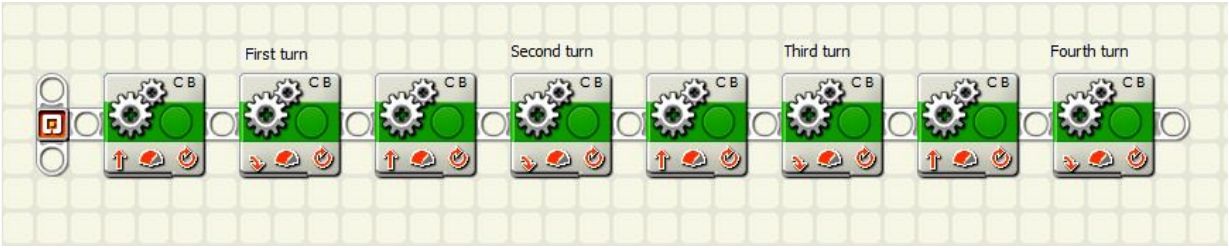


8. Place another move block on the line and set the rotations far enough to get to the other side of the ruler with room to spare.



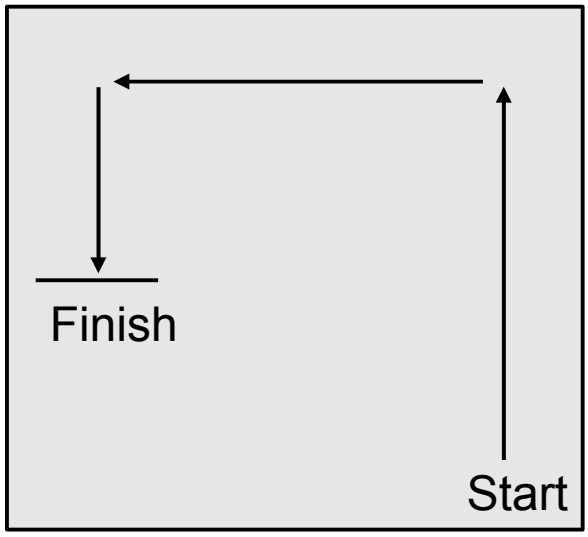
This is the fourth turn that should get the robot back to the starting point and facing the way it was when it started. Some of the turns may need to be adjusted to get it back where it should be.

9. The finished program should look like this.



Secret to success: You must not touch the ruler or you have to start over again, but remember something. There is an old saying the way you don't hit something is to never go near it. You don't need to be close to the ruler when you go around it. Stay a good long way from it and your programming will go much quicker.

2 Turns and Bump



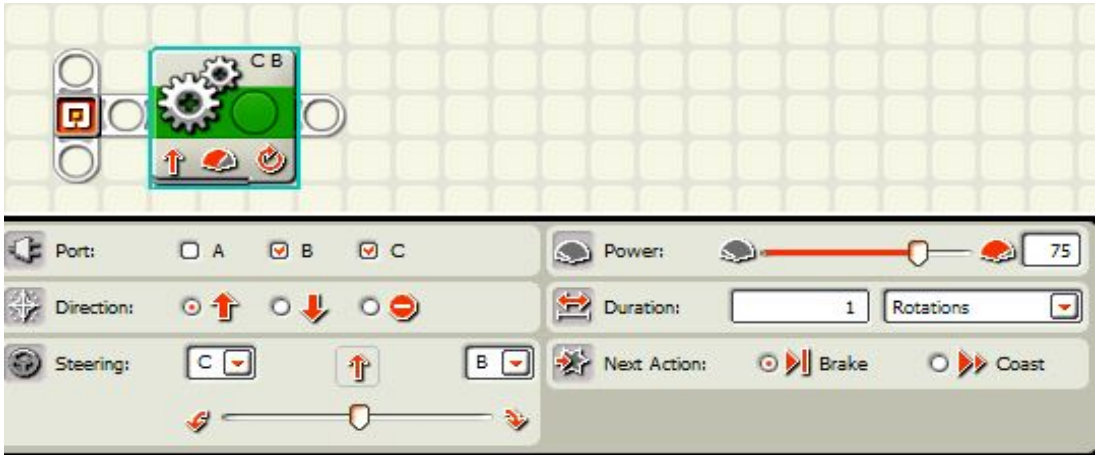
Mission: the robot will follow a course where it will travel forward, turn left, turn left again, and knock down a block.

Equipment
blue painter's tape to make a line on the field.
small block for robot to tip over

Sensors:
none

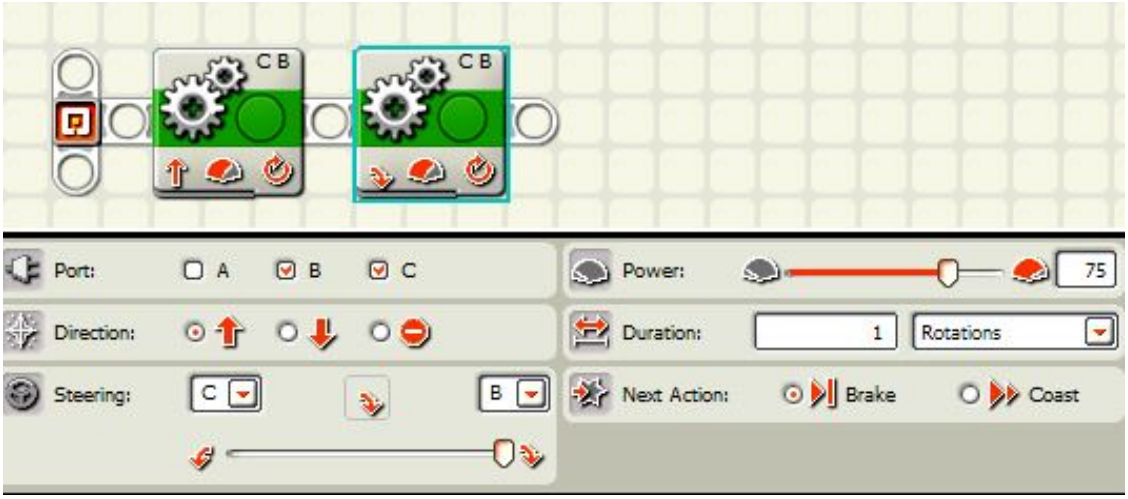
Directions:

1. Place a move block at the start of the program bar. Set the number of rotations to move the robot the distance it needs to go.



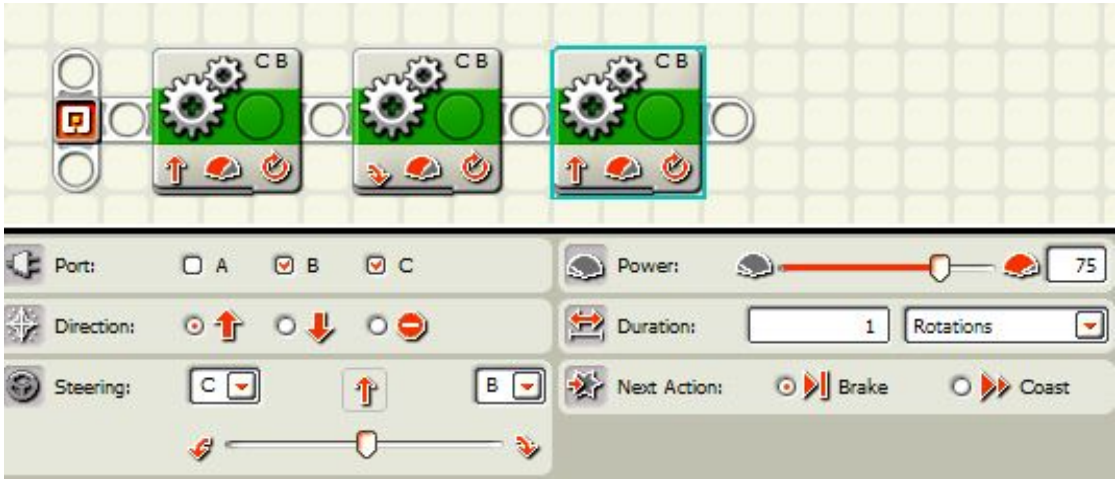
The move block makes the motors rotate. Use trial and error or refer to your notes about the number of rotations needed to go the distance.

2. Place another move block after the first one. Slide the steering bar on the lower left of the illustration in the direction you want it to turn. Set the rotations so that the robot will turn 90 degrees.

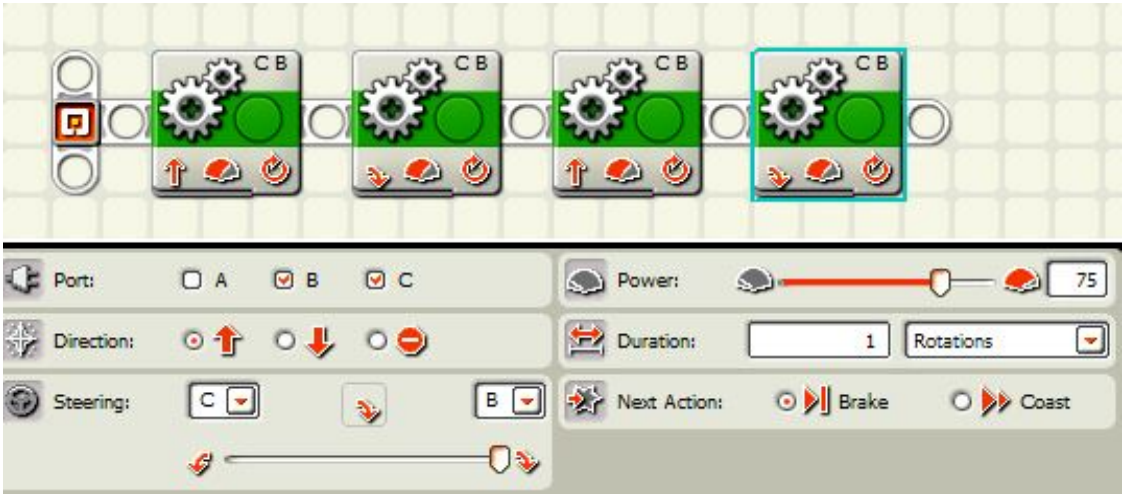


Sliding the steering bar all the way to one side or the other makes the robot turn in place with one wheel turning forward and the other wheel turning backwards.

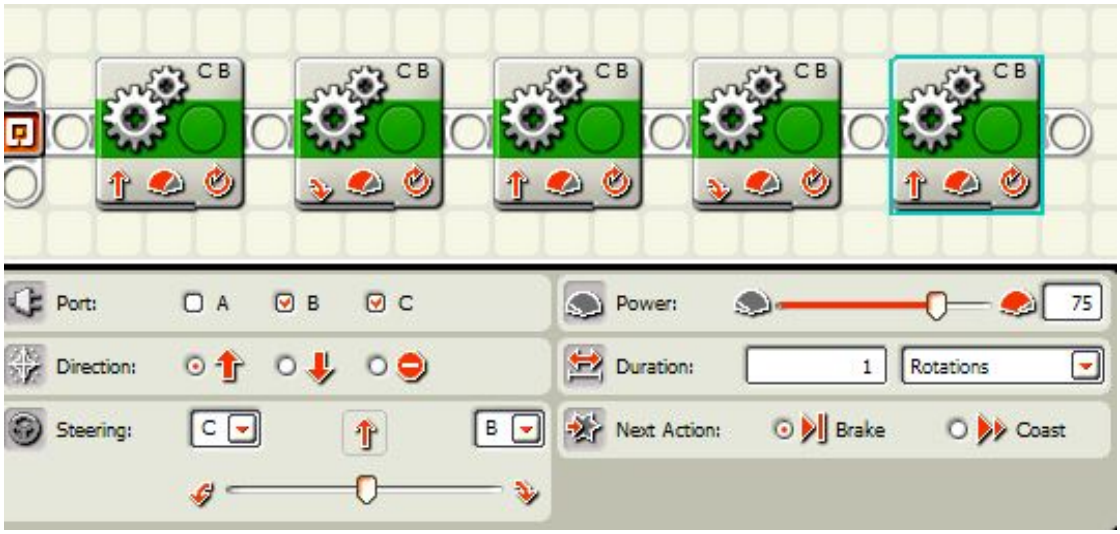
3. Place another move block after the turn and set the rotations for the distance needed.



4. Place another move block and set the steering bar all the way to the side. Set the rotations to make a 90 degree turn.



5. Place another move block on the program bar and set it to unlimited.



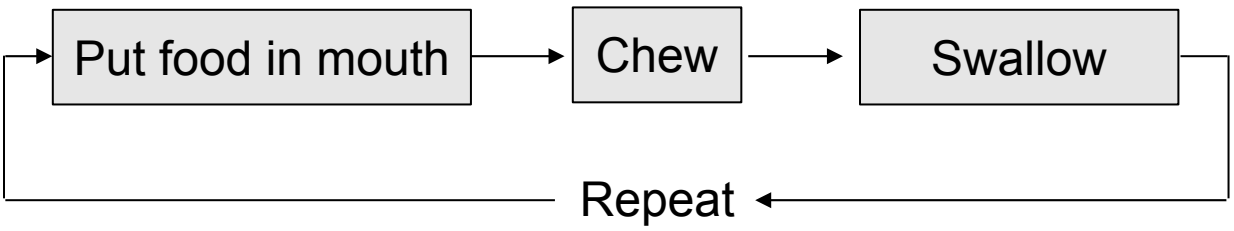
This will make the motors turn until it bumps the block. You will need to adjust the number of rotations to get it to the block and bump it to knock it down.

Secret to success: Use your notes to know how many rotations it takes to go a foot and how many rotations it takes to make a 90 degree turn and it will help you get the rough distances and turns for your mission. A little adjusting will be necessary, but that won't take as long as it would without using your notes.

Loop Blocks,

An Explanation

Loop blocks are for repeating an action over and over and over. Often, we repeat the same thing many times over For instance, when we eat we do this:



Rather than programming a long line of blocks to do this action, you can program a set of blocks that do the action one time and then place it all in a loop so it will repeat. Loops can go on for ever or be triggered to stop by one of the following things:

1. Forever: the loop will run until the dark button on the brick is pushed ending the program.
2. Sensor: Some sensor is triggered. Loops can be triggered by any of the sensors. More on that later.
3. Count: The number of times the robot executes the series of blocks in the loop. This does not mean counting in the sense of time. The number of counts is not a number of seconds.
4. Time: The number of seconds the loop should run. This can be useful, but realize that if the battery is well charged, it may travel farther than when the battery is worn down so the distances traveled may be different as the batteries wear down or get charged.
5. Logic: the loop is told to stop by a logic command. This will not be covered in this book.

The next mission will use a Loop block.

Ruler with Loops

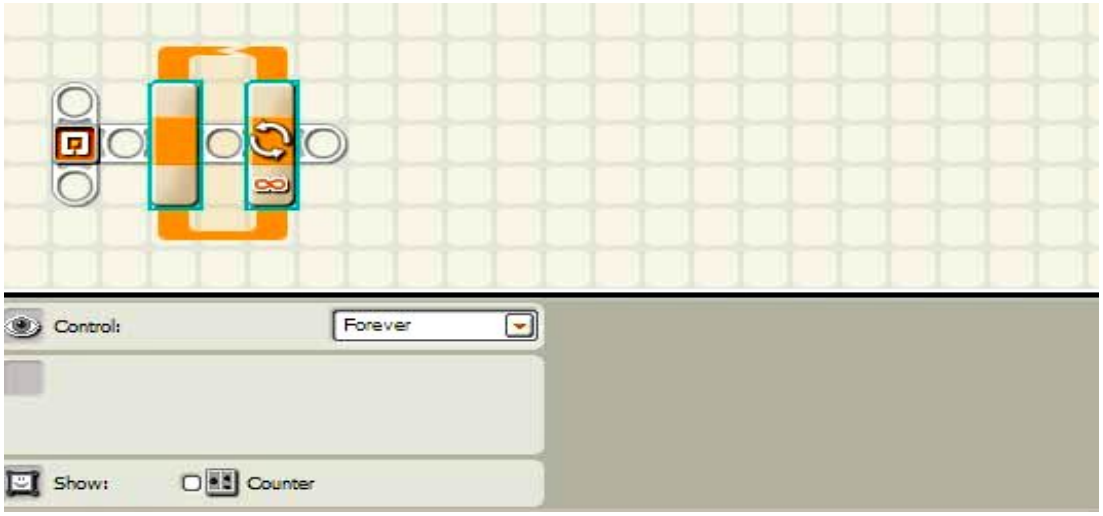
Mission: the robot will drive around the ruler without touching it using loops in the program to use less programming blocks.

Equipment:
robot

Sensors
none

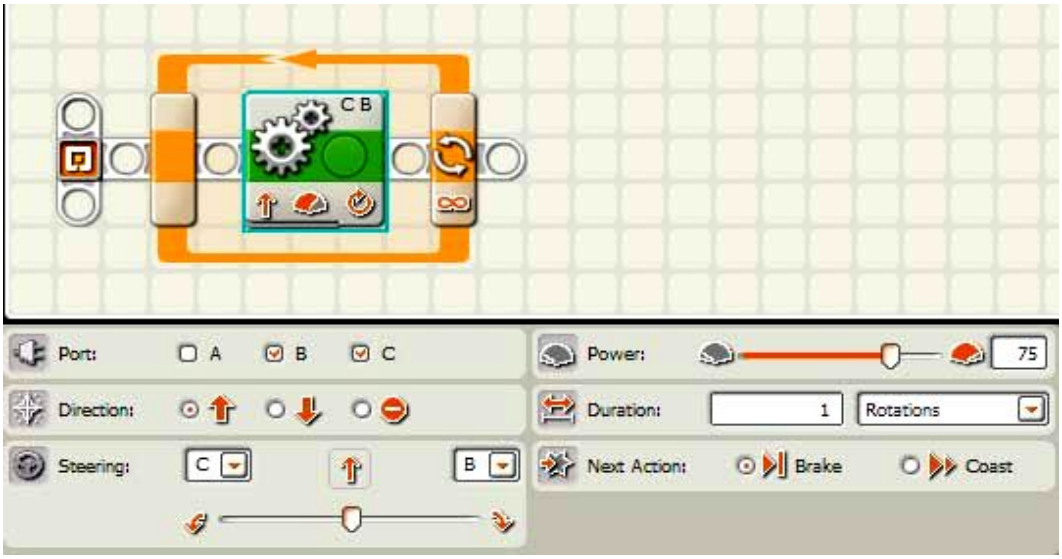
Directions:

1. Place a Loop block on the program bar.



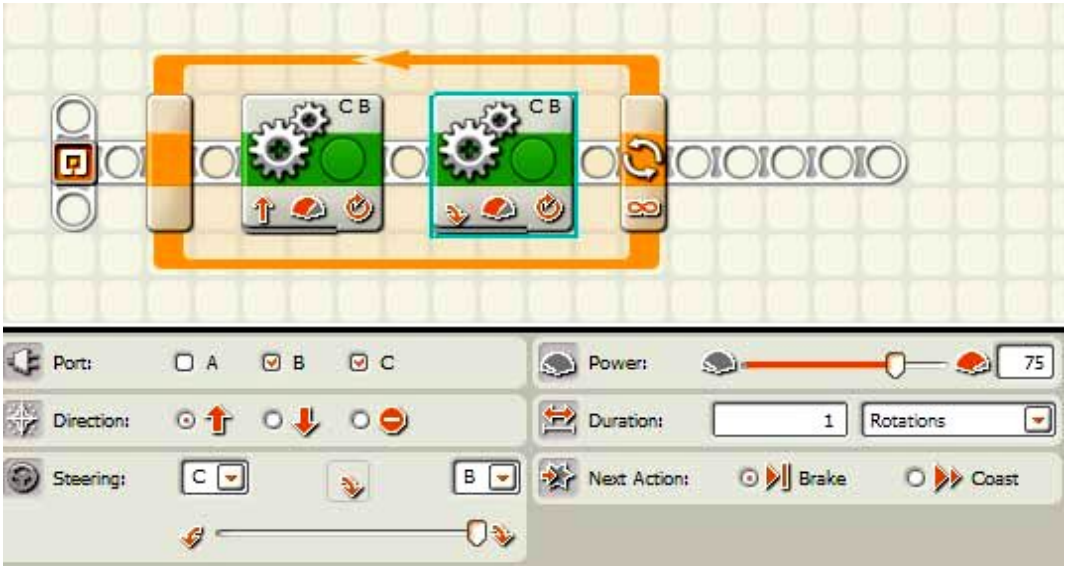
This loop will repeat over and over the program blocks you will put inside it.

2. Place a Move block into the Loop. Hold it over the middle of the Loop until the Loop opens up wide enough to put it in. Set the rotations of the Move block high enough to travel far enough to go past the ruler.



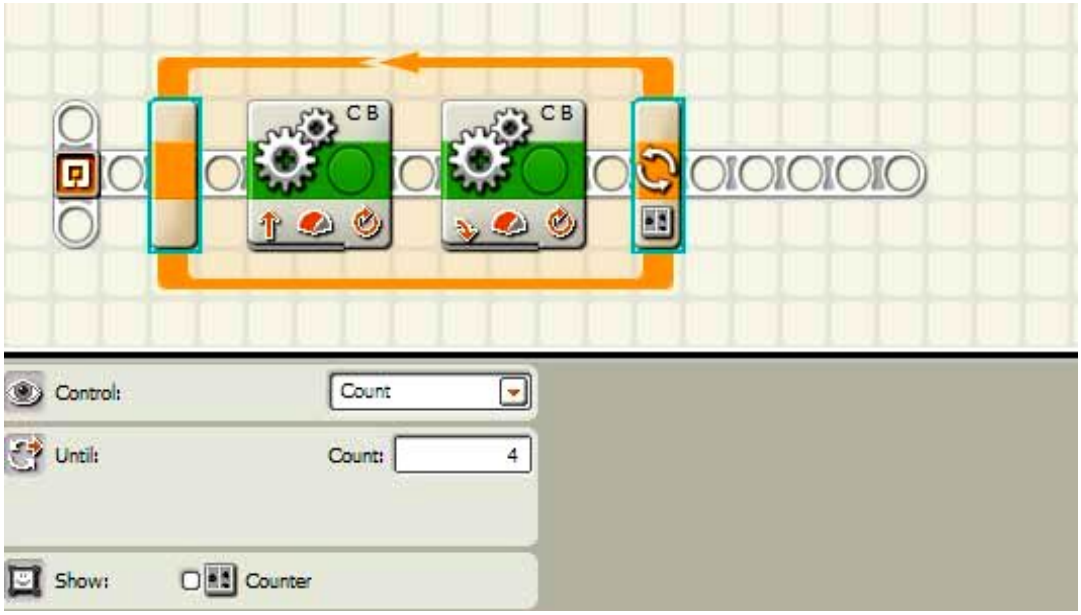
The move block will make the robot go forward past the end of the ruler.

3. Place another Move block after the first one and set it to move all the way to the right on the Steering setting at the bottom left of the control panel.



This will make the robot turn. Adjust the rotations so the robot turns in a 90 degree angle.

4. Click on the loop and set it to Count and set the count to 4. In other words, the Loop goes through the two Move blocks 4 times.



Setting the count on the loop to 4 will make the robot go up one side, turn, go past the end of the robot, turn, go down the other side, turn, head back to the starting place, and turn to face the way the robot started.

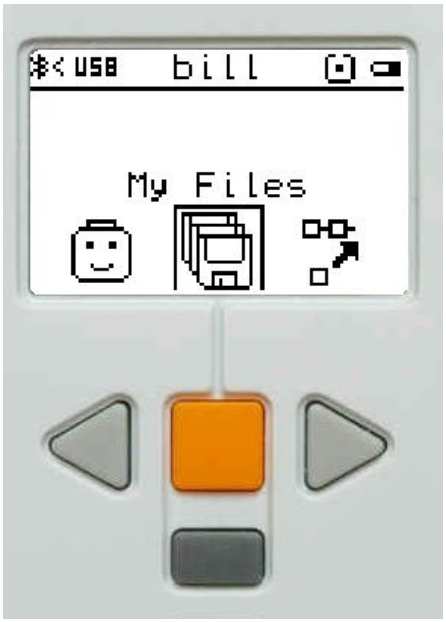
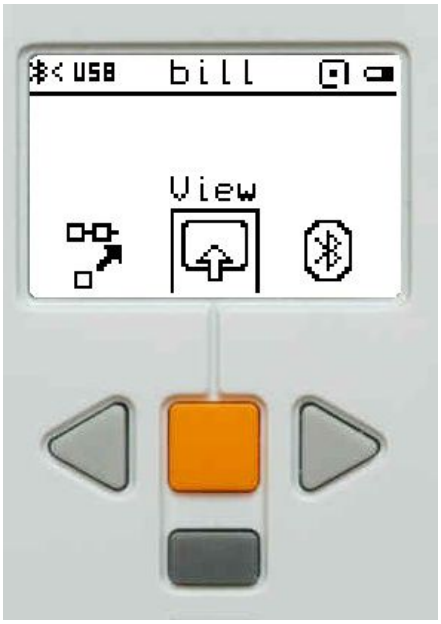
Secret to success: Use loops whenever you can because it will save you a lot of time programming. It will also keep your program a little shorter and easier to handle.

Checking Sensors and Wheel Rotations

As you are learning how to program, it helps to be able to test the sensors to make sure they are working or if there is a problem. That way, you can see if your program has a problem or if the sensor or cord is having a problem.

Be sure to have the sensor plugged into one of the ports of the intelligent brick and notice what port the sensor is plugged

First, this is how you get to the sensor view part.

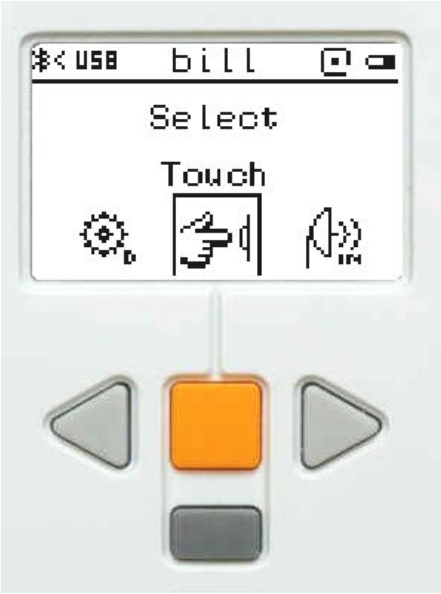
1. Start with the level that says My Files. 	2. Scroll to the right by pushing the right gray button until you see View. Push the center orange button.  *This will open up the view section so you can check the various sensors as well as the rotations of the motors.*

Note: The program used to make the illustrations of the intelligent brick is freeware program called NeXTScreen, and is found at http://bricxcc.sourceforge.net/utilities.html. It shows the screen of the NXT intelligent brick on the screen of your computer.

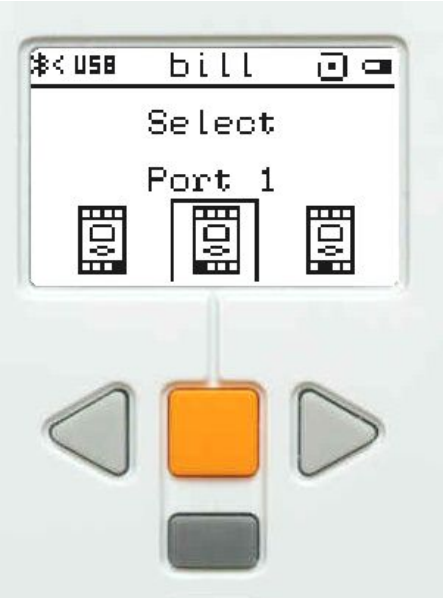
Touch Sensor

Once you get into the View section, you can use it to get readings on various sensors and motor rotations. Here is how you check the touch sensor. Be sure it is plugged into a port.

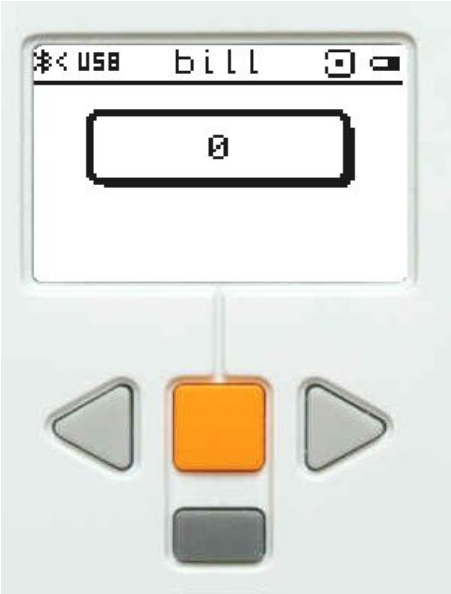
1. The touch sensor looks like this. Push the center orange button to access it.



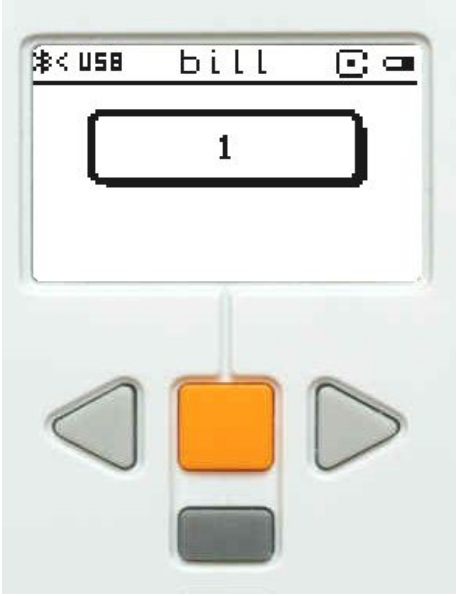
2. The touch sensor is usually attached to Port 1.



3. The touch sensor will say 0 if it is not pushed.

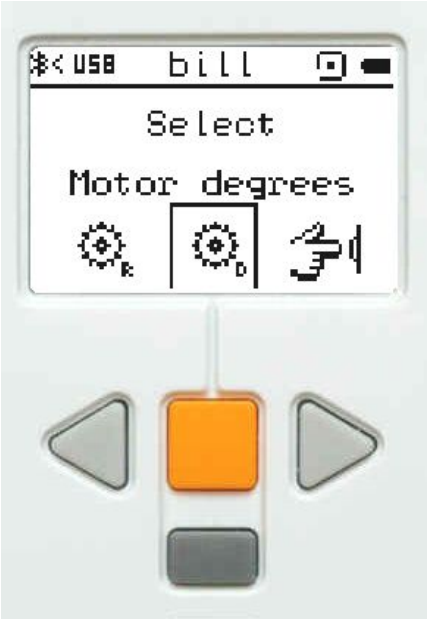


4. It will say 1 if the touch button is pushed.

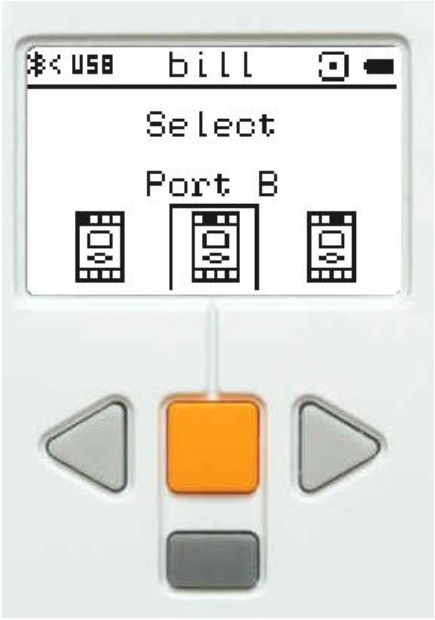


Motor Rotations

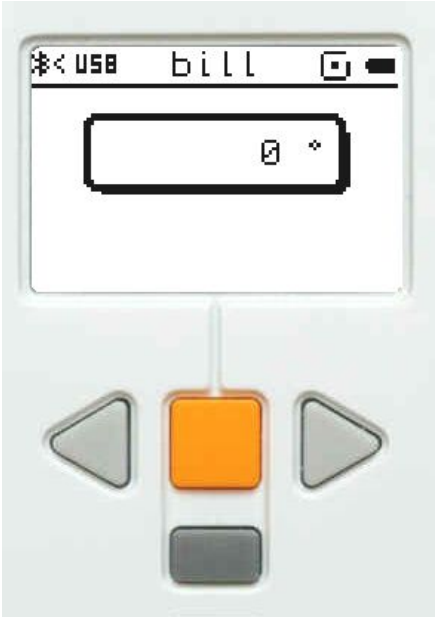
1. Go to Motor degrees and push the orange button.



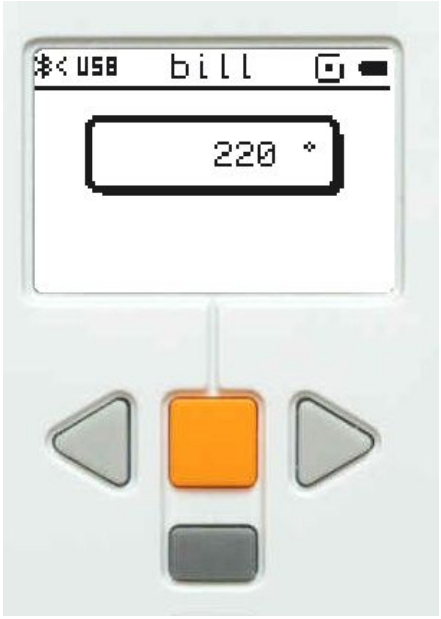
2. Use the side buttons to pick a port connected to the motor you are monitoring.



3. It starts out at zero.



4. The number goes up as the wheel turns forward.

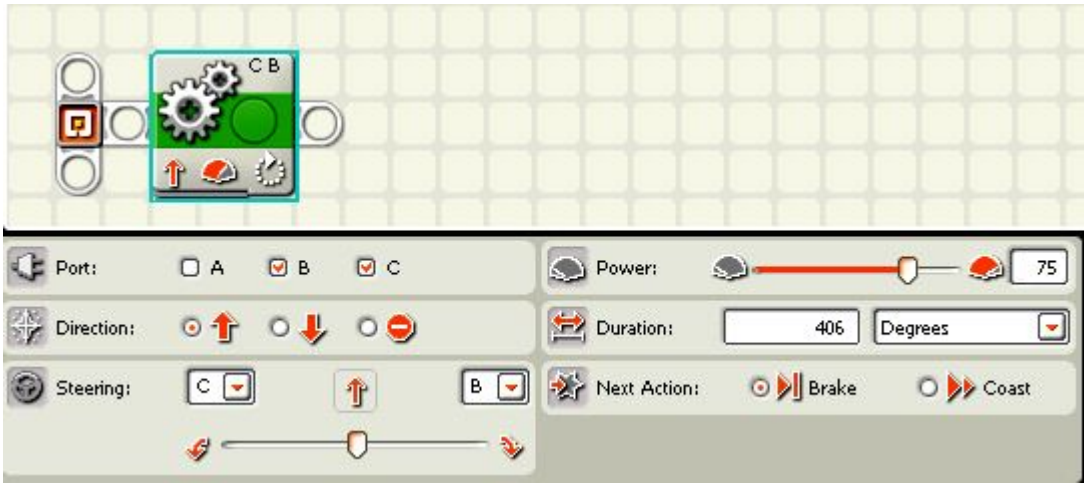


Remember that 360 degrees equal one rotation.

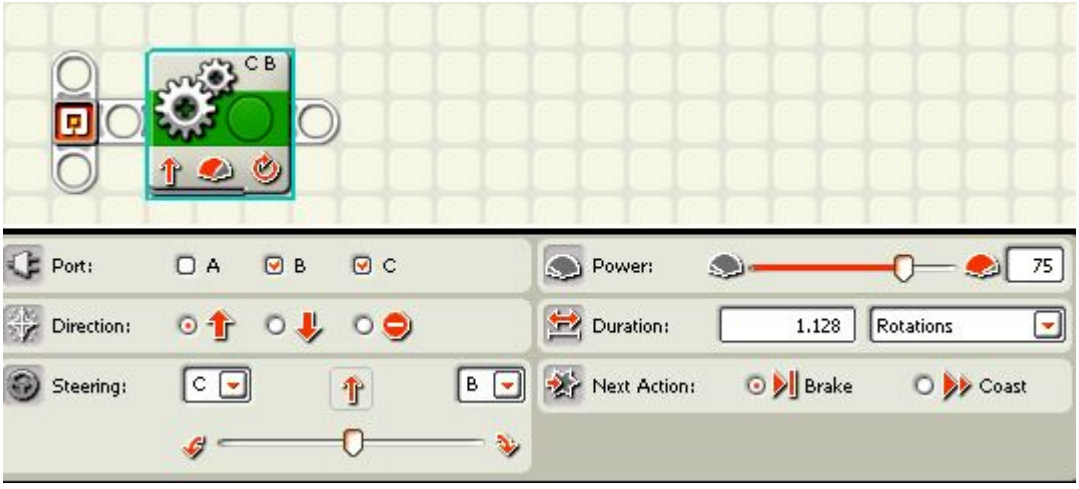
Changing degrees to rotations

When reading the degrees a motor has turned, it can be a pain to do the math to change it to rotations. Lucky for you, the Mindstorms program can do it automatically. You can also just leave it as degrees, but if you prefer rotations, you can have rotations by following the steps below.

Set the rotations drop down menu to degrees. Type in the number of degrees you measured.

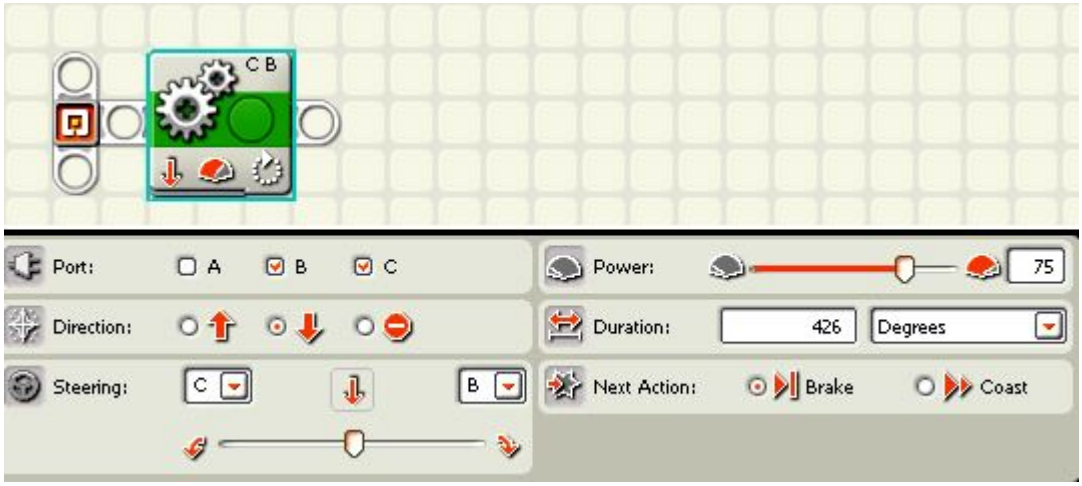


If you like working with rotations, you can change the degrees back to rotations by changing the drop down menu.

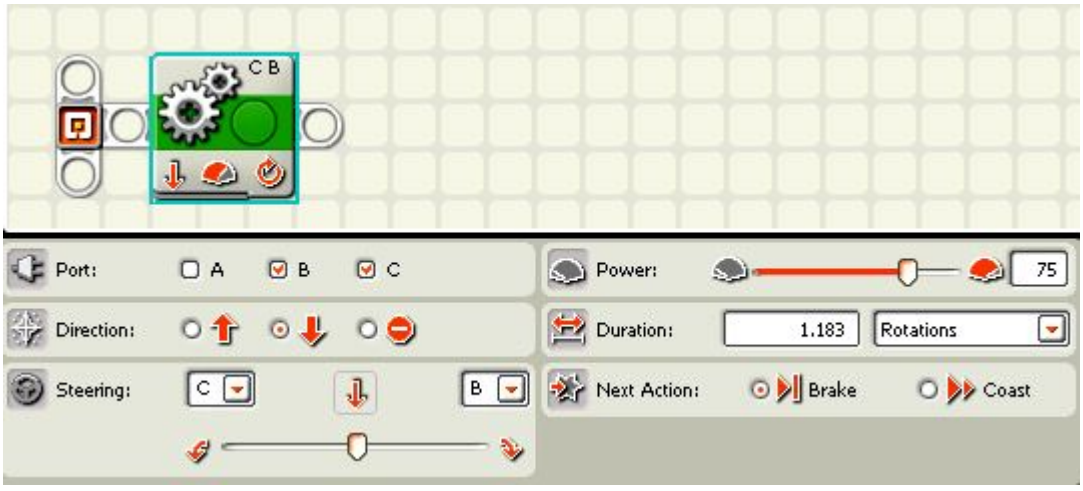


Dealing with negative numbers

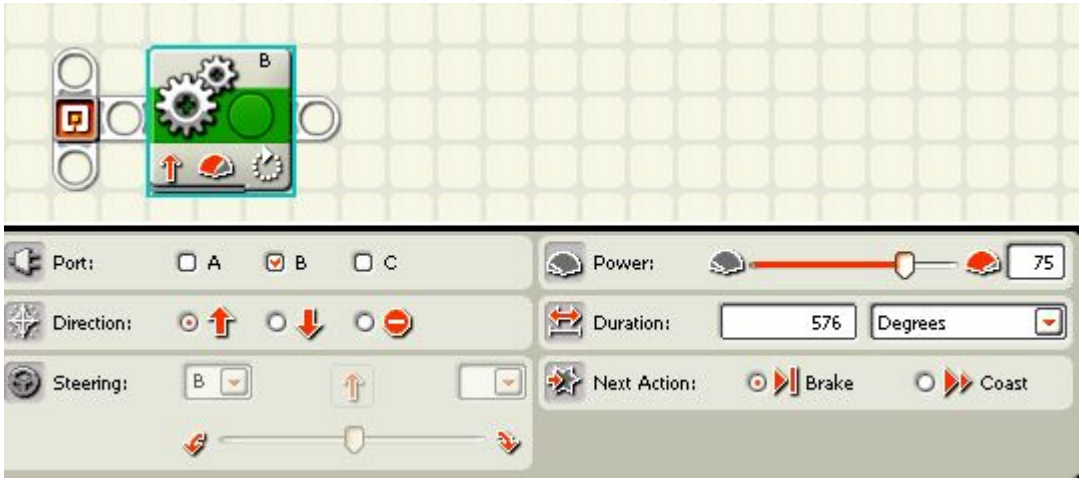
If you pulled the robot backwards instead of pushing it forward, you can get a negative number. Since the move blocks do not use negative numbers, you can use the negative degrees by changing the robot to go in reverse by clicking on the arrow pointing down and by changing the negative degrees to positive degrees.



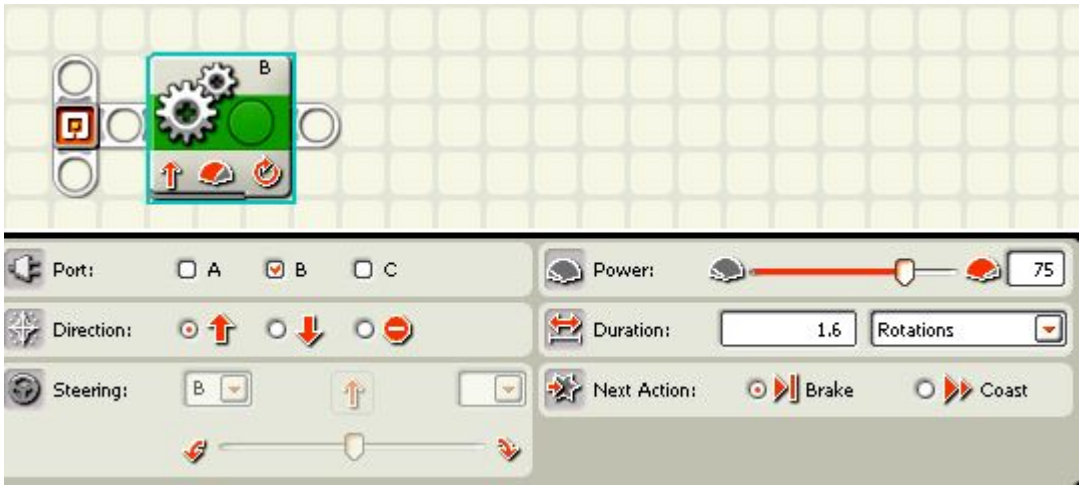
And again, if you want to change it to rotations, you can do that by just changing to rotations on the drop down menu.



You can also just program one wheel for a turn. Here one wheel is going to rotate to make the robot turn. Click on C so it is no longer selected. Change the move block to degrees. Type in the number.



And you can change it back to rotations if you like. You don't have to change it if you don't like it that way.



Go Hit Back

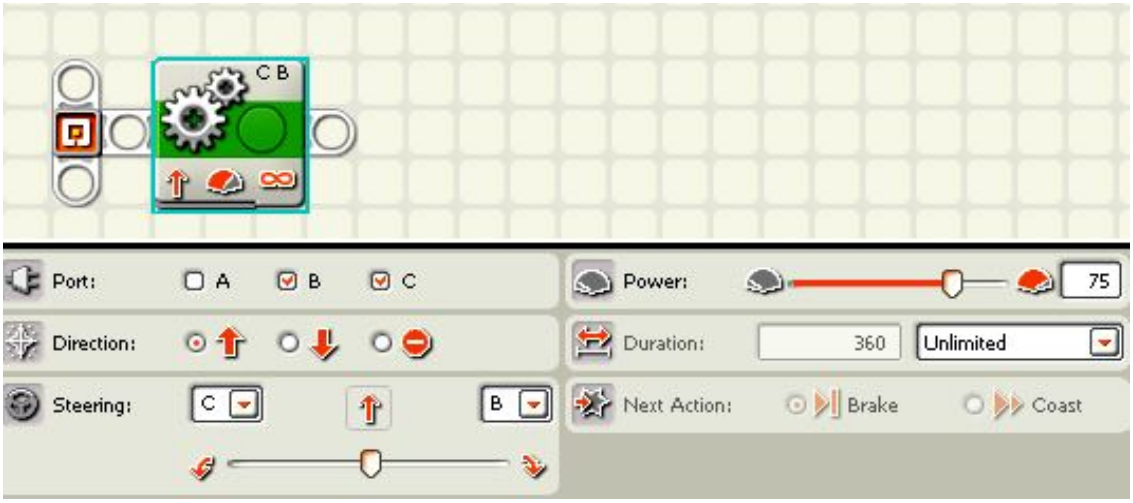
Mission:
The robot go forward until it hits the wall and then it backs up one rotation.

Equipment:
none

Sensors:
touch

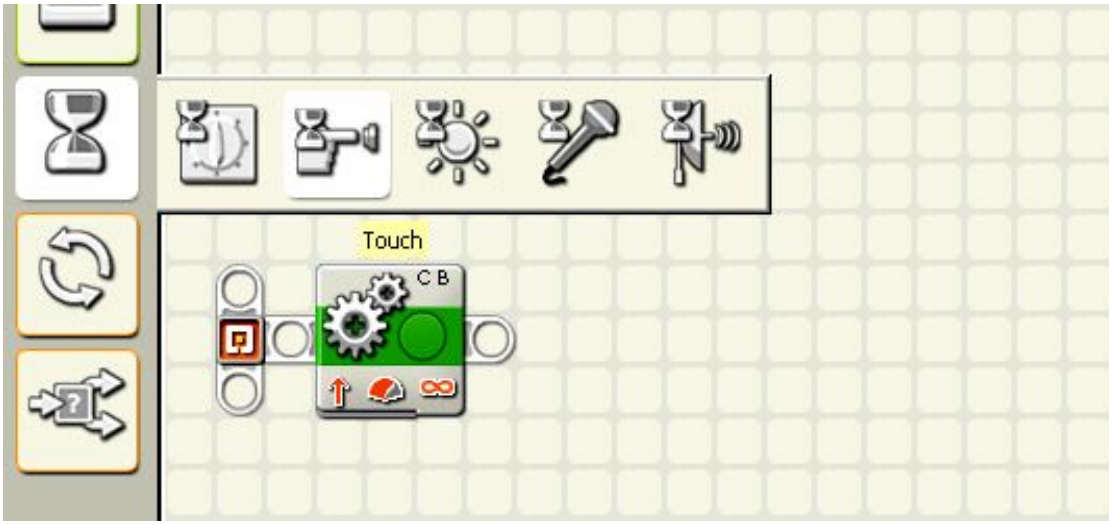
Directions:

1. Place a move block and set the duration to unlimited. It is found on the lower right side of the illustration.

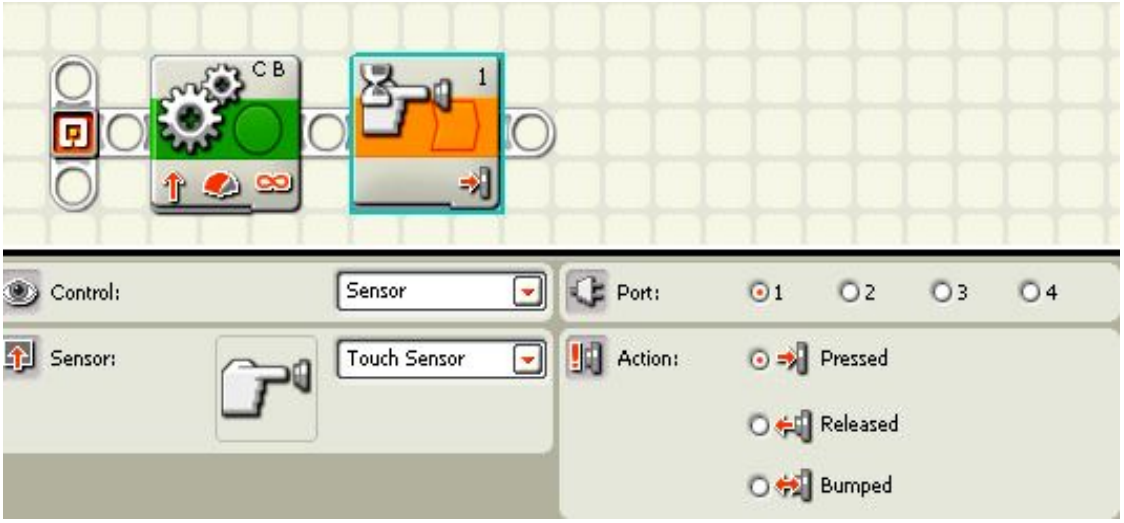


Unlimited will make the motors turn and not stop until something tells it to stop. The command to tell it to stop will be the next part of the program.

2. Place the cursor over the wait block and it will open up to show you several choices. Click on the touch wait block. It looks like a finger touching a button.

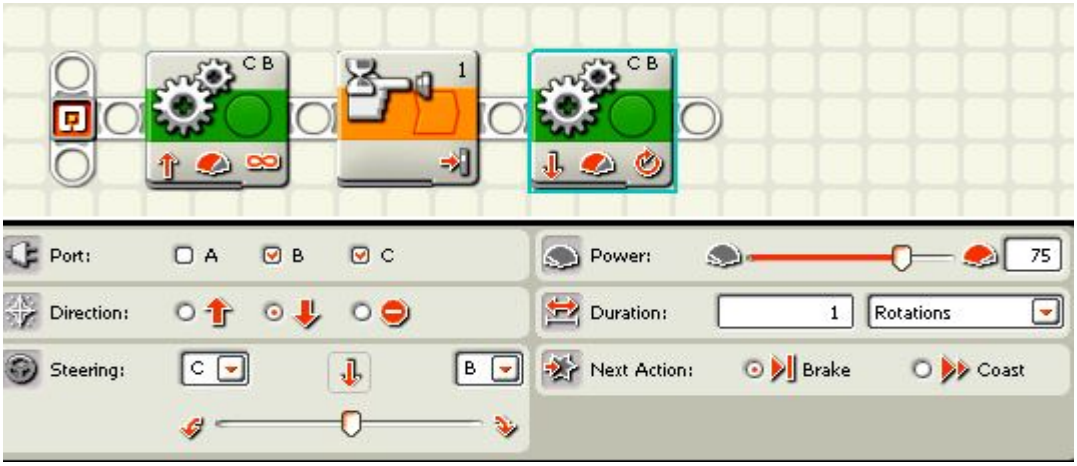


3. Place the wait block on the program bar. Notice it is set for port 1. Be sure your sensor is plugged into port 1 or that you change the port to whichever port you have the sensor plugged.



This tells the robot to keep going until something presses the touch sensor. When something touches the touch sensor, this block will let the program move on to the next block.

4. Place a move block on the program bar and set if for reverse. Leave the rotations set at 1.



This block makes the robot go in reverse. So when the robot touches something, the touch block lets the program move to the next step which is to go in reverse.

Touch Sensor

The touch sensor has three actions:

1. Pressed means that the button is pushed and not released for at least a second.
2. Released means that the button was already pressed and now it is released.
3. Bumped means that the button is pushed and then released in less than a second.

Pressed

You will use pressed the most since you will want the robot to hit something like a wall and then do something different. The robot will roll up and push against something for a second and then it will do the next action.

Released

This could be used if the robot has the touch sensor already pushed and something releases the pressure. You could program the robot to push against a block, then, when the block is removed, the robot will do something else. Released is not used very much.

Bumped

This could be used when the robot would go by something that would push the button on the touch sensor and then release it without the robot doing anything new. It has to be touched and released in less than a second. Again, this action is not used very much either.

Head Banger

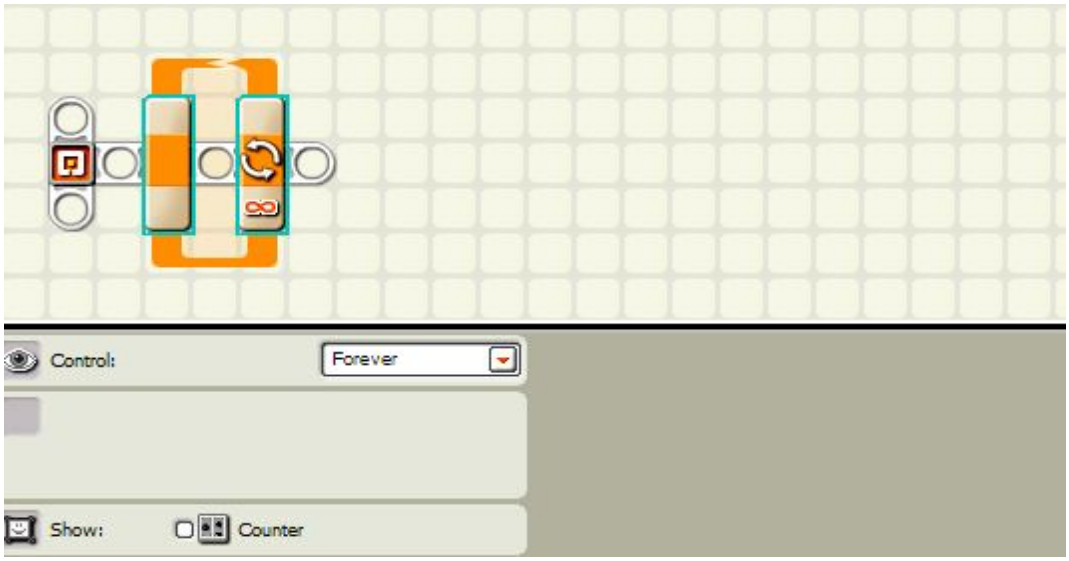
Mission:
The robot will move forward until it touches wall with touch sensor. It then backs up 2 rotations and repeats the actions for a total of three times.

Equipment:
none

Sensors:
touch

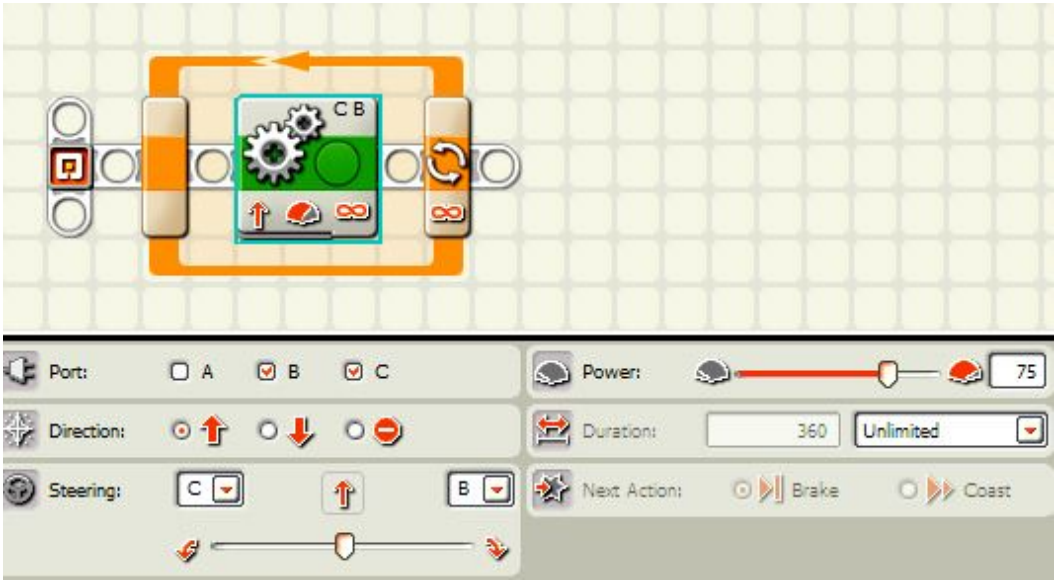
Directions:

1. Place a loop in the program bar. Leave setting as forever.



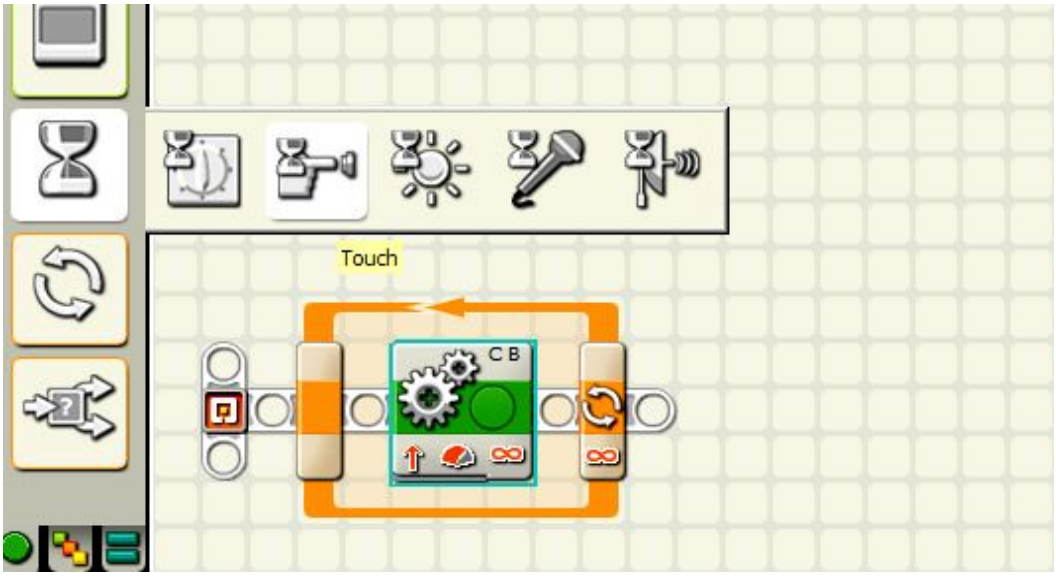
This will make the program blocks on the inside of the loop repeat over and over again.

2. Place a move block inside the loop. Drag it over the loop and the loop will open up to allow it to be put inside. Set it to unlimited.



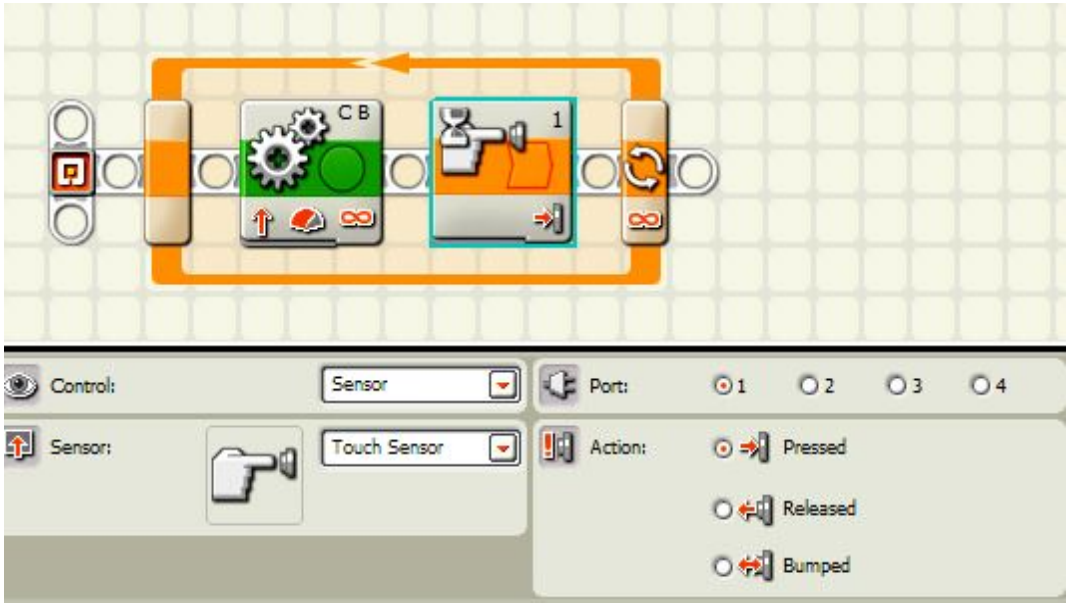
This makes the robot move until the touch block (next) bumps into something.

3. Place the cursor over the wait block and it will open up to show the choices. Pick touch.



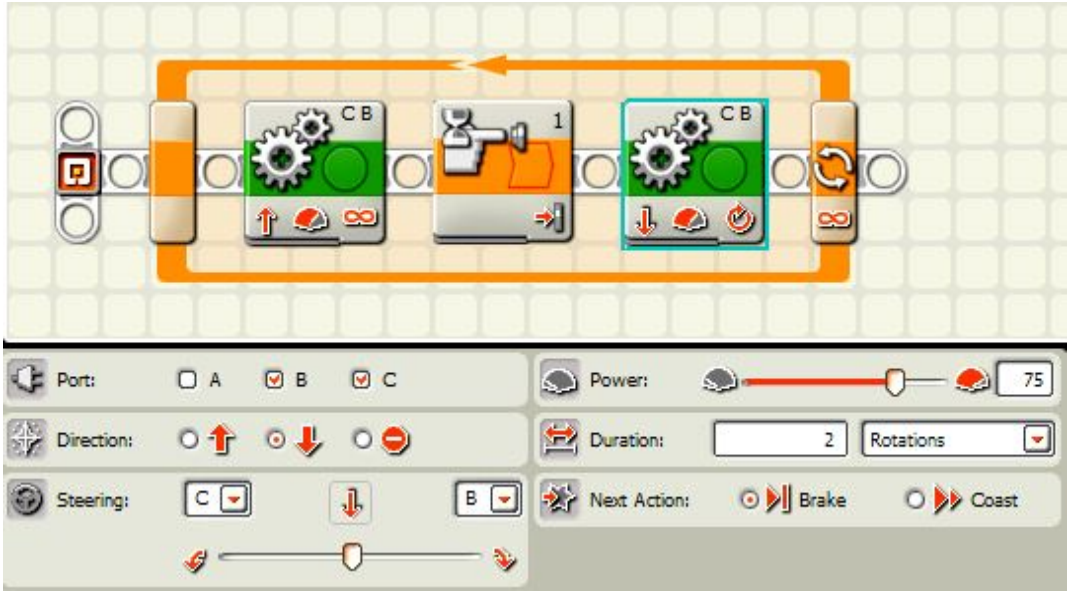
This allows you to have a choice of several different types of wait blocks.

4. Place the touch wait after the move block.



The block will make the program move to the next block when the sensor is pushed.

5. Place another move block on the bar and set for 2 rotations and set it to reverse.



The program works like this: first, the move block makes the robot go forward until the touch sensor gets pushed; then the robot goes back two rotations; after that, the loop makes the robot repeat the actions until you push the dark gray button on the robot.

Go Hit Turn

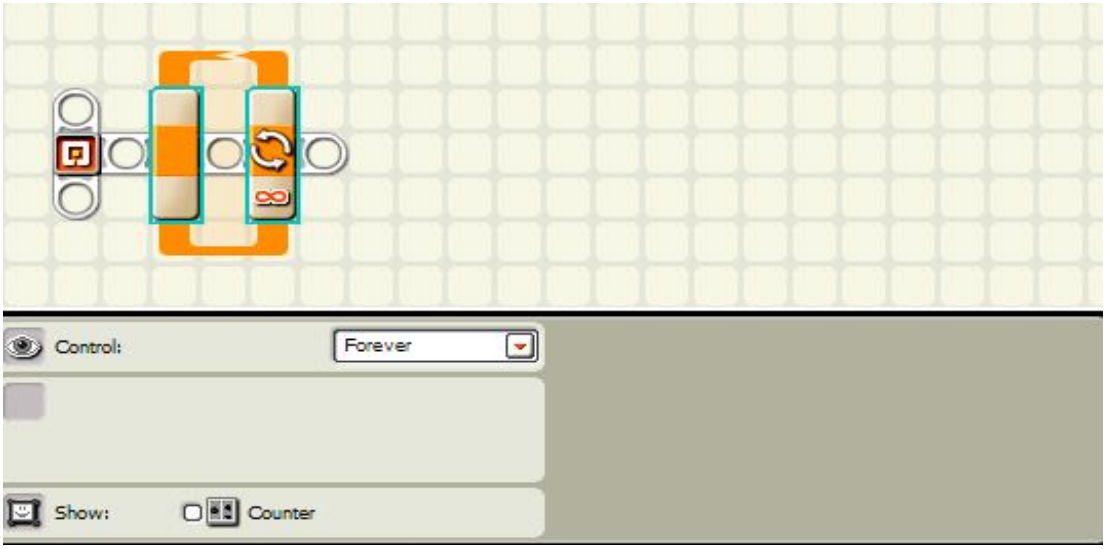
Mission:
The robot will bump into a wall, turn 90 degrees, and go forward again. It will do this until stopped.

Equipment:
White table top or playing field.
Walls (could be 2x4's or books)

Sensors:
Touch

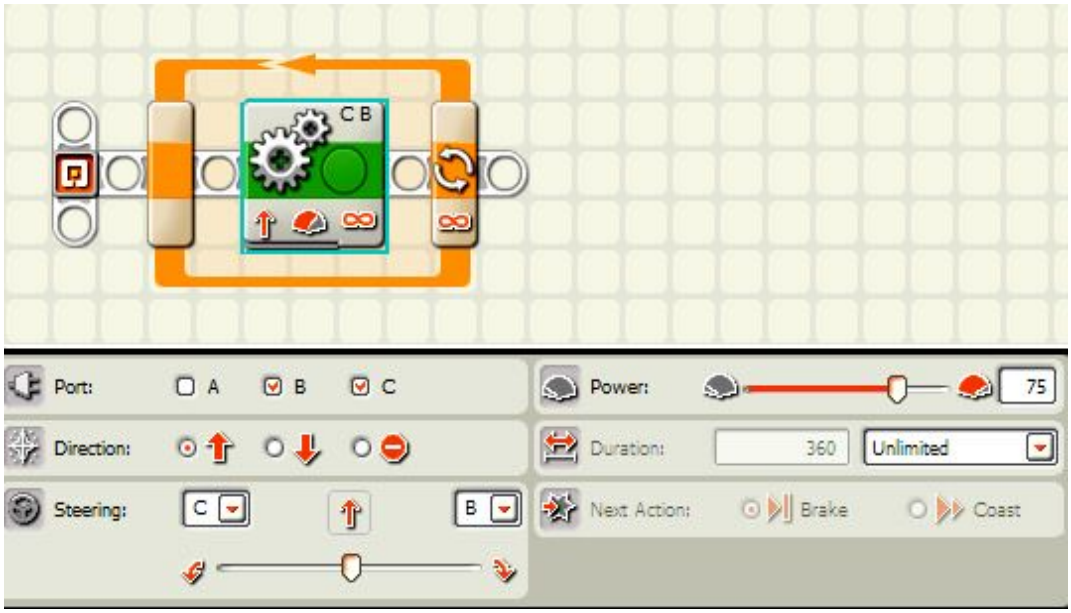
Directions:
Attach the bump sensor so that it faces the front of the robot.
Program the robot so that it will run to the end of the wall, bump the wall, back up, turn to either the right or the left, continue to the next wall and do the same until it has hit all four walls. Have it stop after hitting four walls.
Use a bump switch, move, and a loop block.

1. Place a loop block and leave it at Forever.



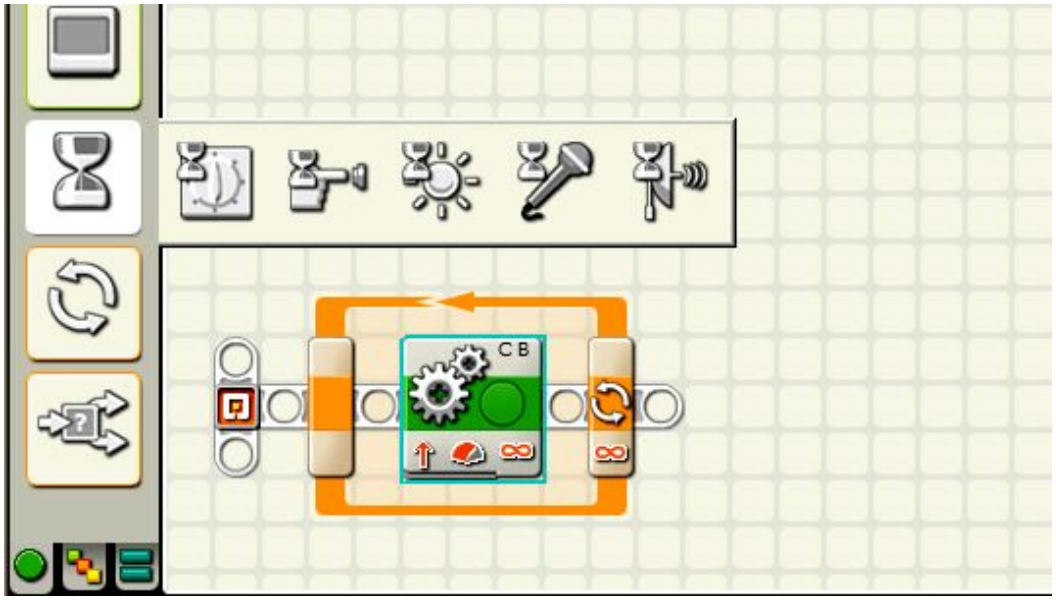
Loops are used to repeat a group of actions over and over. We want the action to be repeated over and over so we leave it at forever.

2. Place a move block inside the loop. Set it to unlimited.

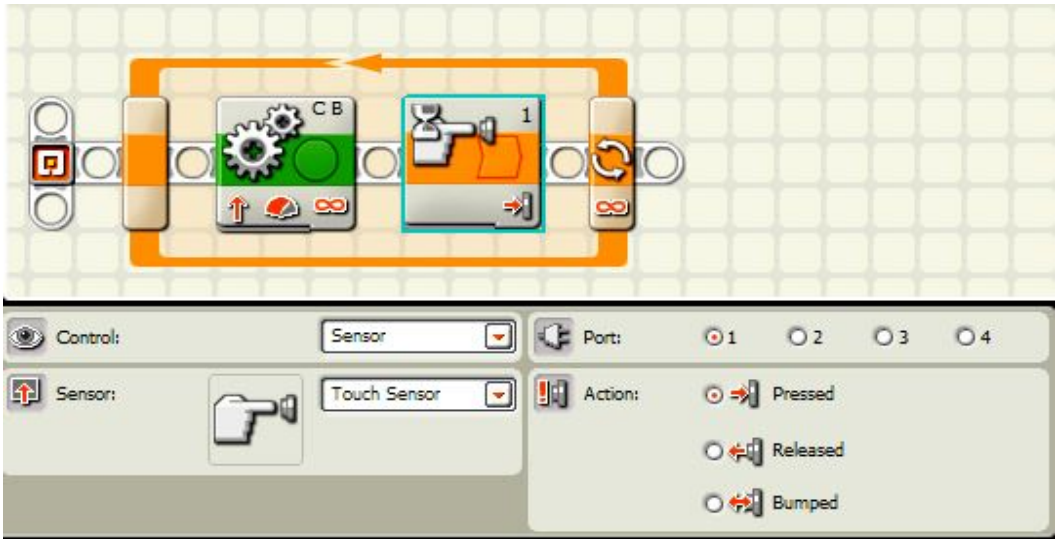


Setting the move to unlimited will make the robot keep moving until it is told to move to the next thing by some type of wait block. That is the next step.

3. Place your cursor over the wait block and it will show several choices. Pick the touch wait block. It looks like a finger touching a button.

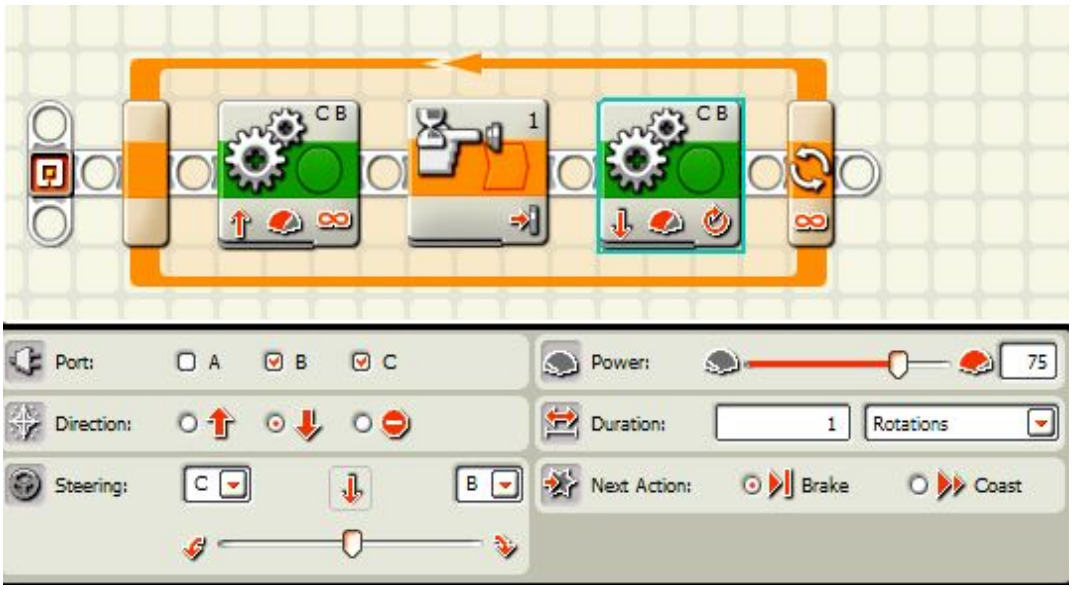


4. Place the touch wait block after the move block.



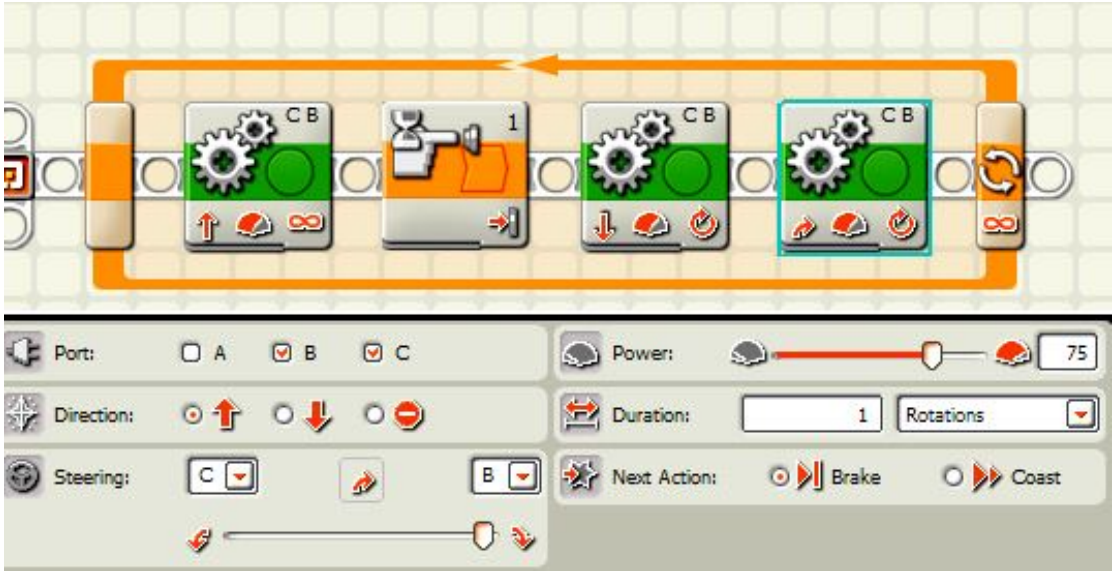
The touch wait block will stop the unlimited move once something presses the button on the touch sensor for longer than a second.

5. Place a move block after the light wait block and set it to reverse by clicking the arrow that points down in the left middle area.



The move block set to reverse makes the robot go backwards for one rotation of the wheels.

6. Place a move block after the last move block and slide the steering bar all the way to one side. This will cause the robot to spin in place when it turns. You can adjust the turn to be as narrow or as wide as you like. Maybe set it at a 90 degree turn and see if you can program it to touch near the four corners of the practice table.



This move block makes the robot turn by spinning in place. You can control the amount of turn by setting the rotations more or less until you get the turn you want.

90, 180, 360 Spins

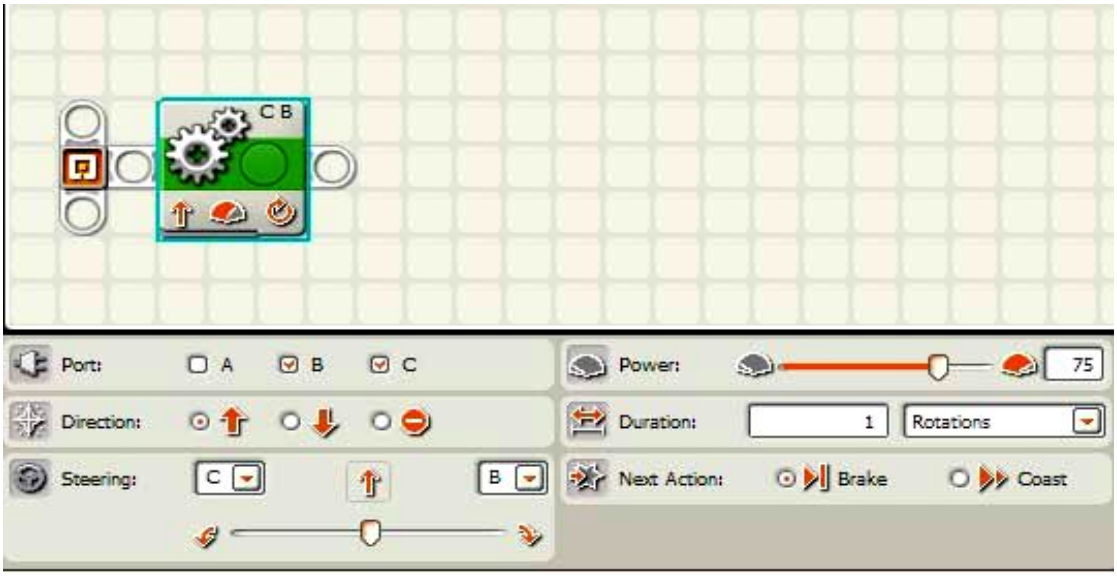
Mission:
The robot will spin in place 90 degrees to the left, then 180 to the right, then 360 to the left again, and stop.

Equipment:
none

Sensors:
none

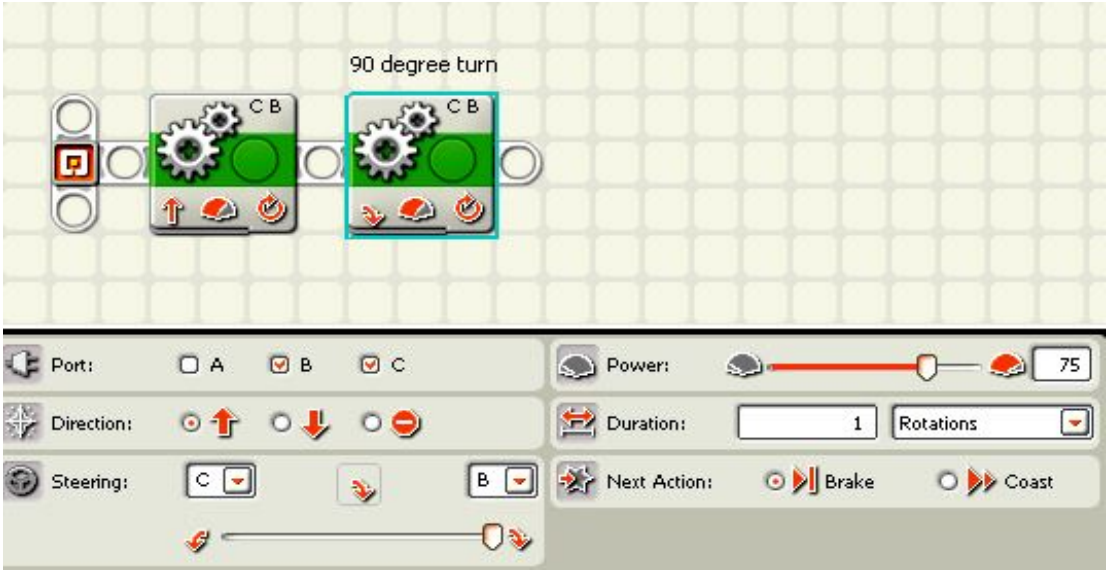
Directions:

1. Place a Move block at the program bar and leave it at one rotation.



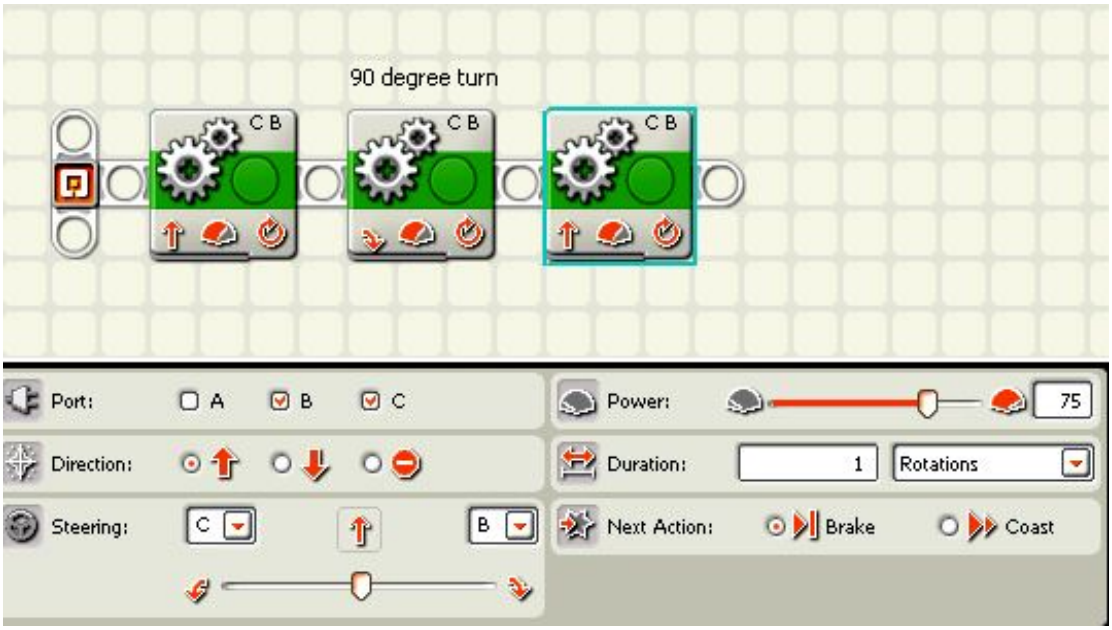
This makes the robot go forward one turn of the wheel.

2. Place another Move block after the first Move block and set it all the way to the right.

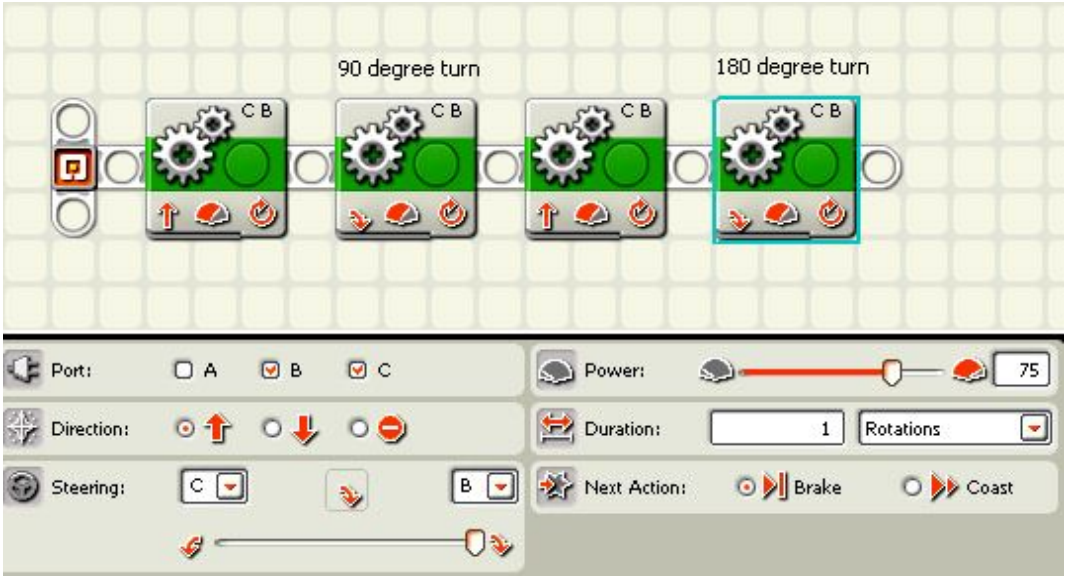


This makes the robot will spin in place. You will need to change the rotations so the robot will make a very accurate 90 degree turn.

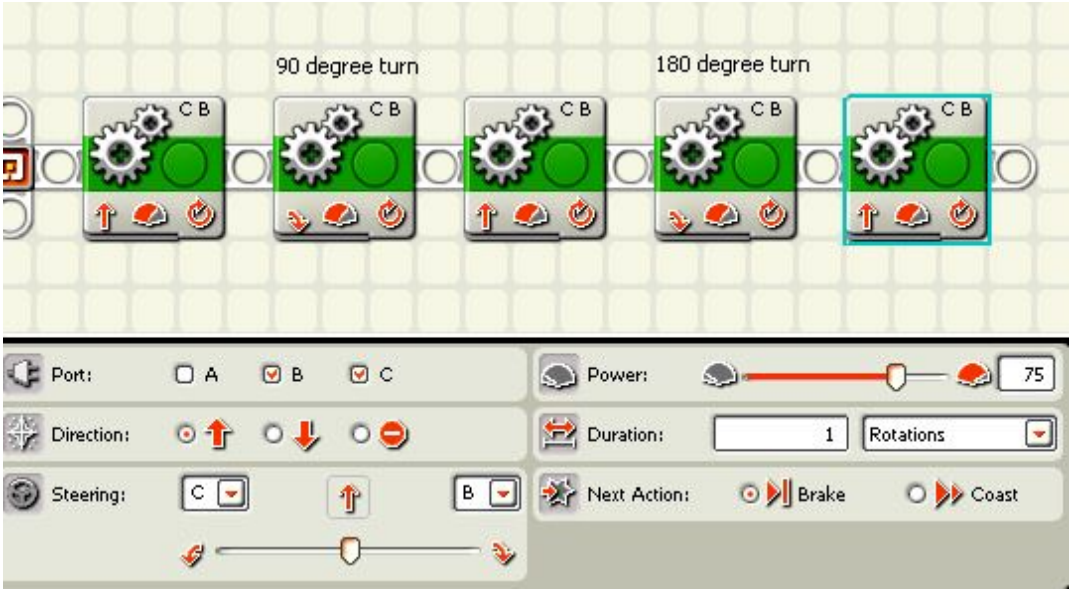
3. Now place another move block into the program and leave at one rotation.



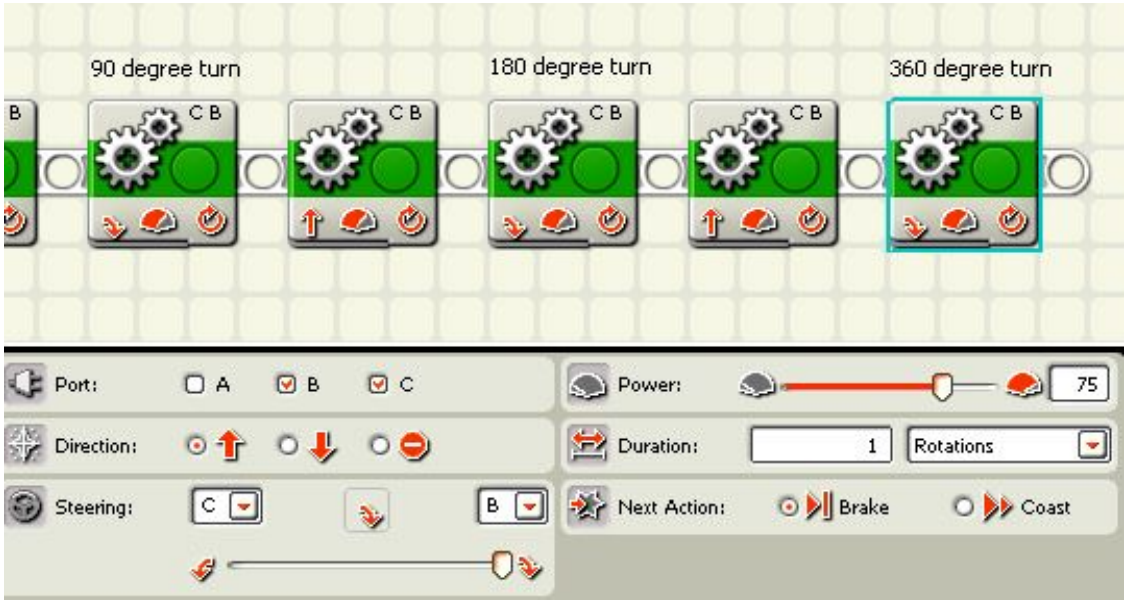
4. Place another move block and set it to double the setting for the 90 degree turn. This should give you a 180 degree turn. You may need to adjust it a bit to make it more accurate. Set the steering all the way to the right.



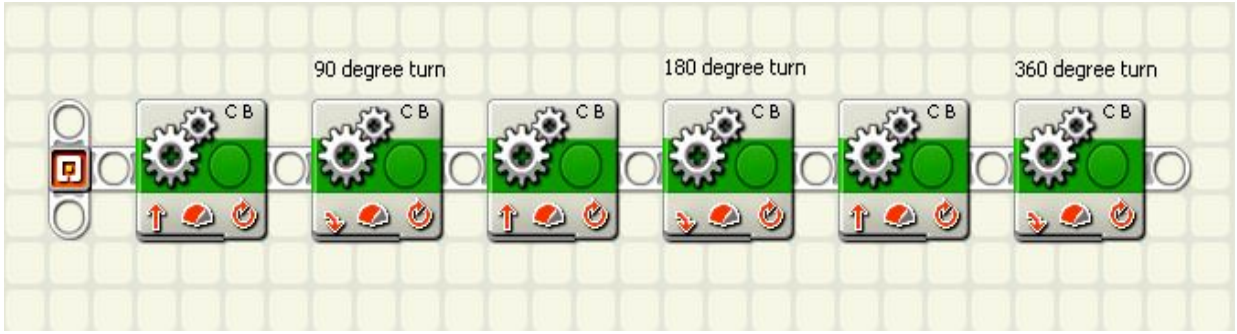
5. Place another Move block and leave it at 1 rotation



6. Place the last move block and set it to rotate double what you did for step 4. This will give you a 360 degree turn. Remember to set the steering bar all the way to the right.



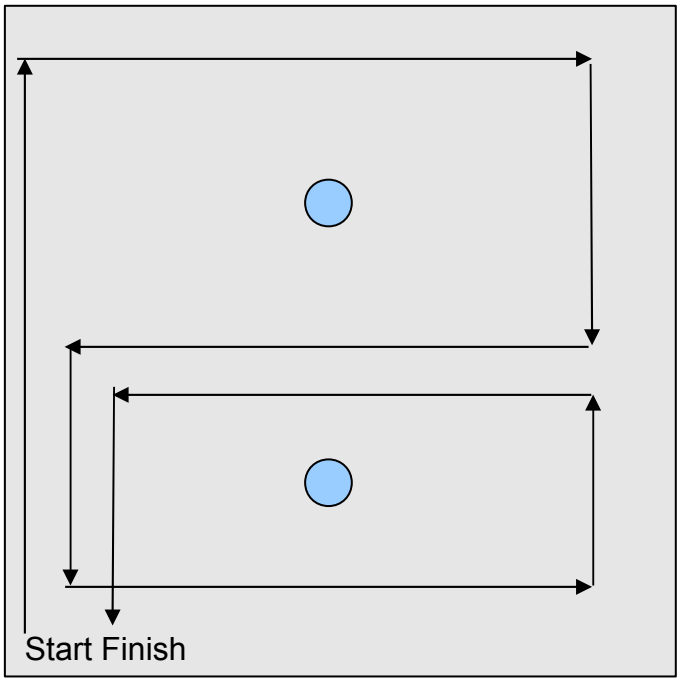
This is what the final program looks like.



Secret to success: As you can see from this exercise, that various turns are easy to figure out once you know how to make a really accurate 90 degree turn. You can double it for a 180 degree turn or you can cut it in half for a 45 degree turn. Another secret is to use as few turns as possible since every time your robot turns, there is a chance that it will not turn perfectly and over several turns, this can make the robot get out of position quite a bit.

Figure 8

Mission: the robot will do a figure 8 around the course using a combination of straight runs and pivot turns.

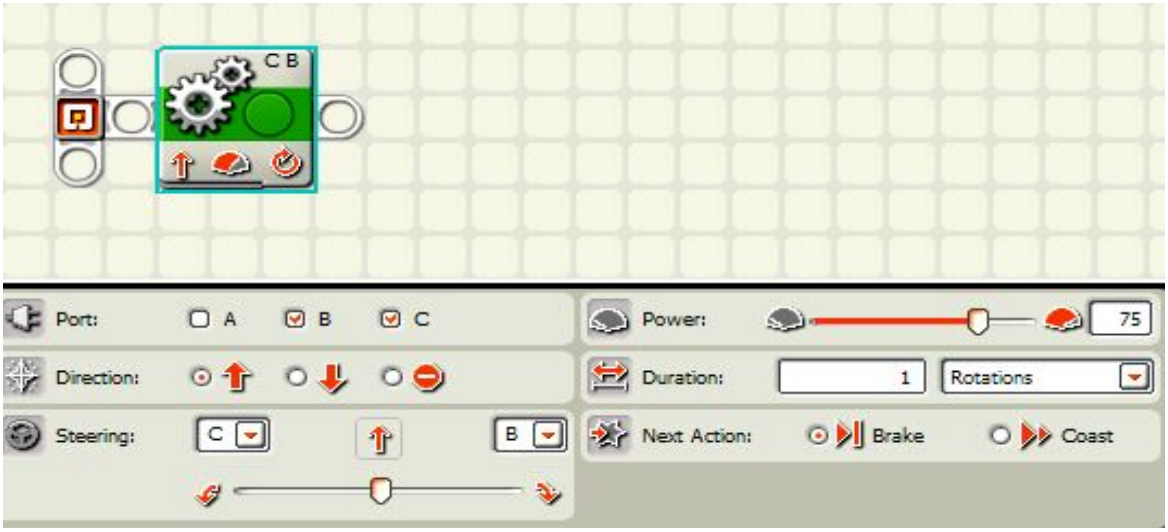


Equipment:
2 empty 16 ounce soda or water bottles to use as posts for the robots to go around. Set them up as shown in the illustration.

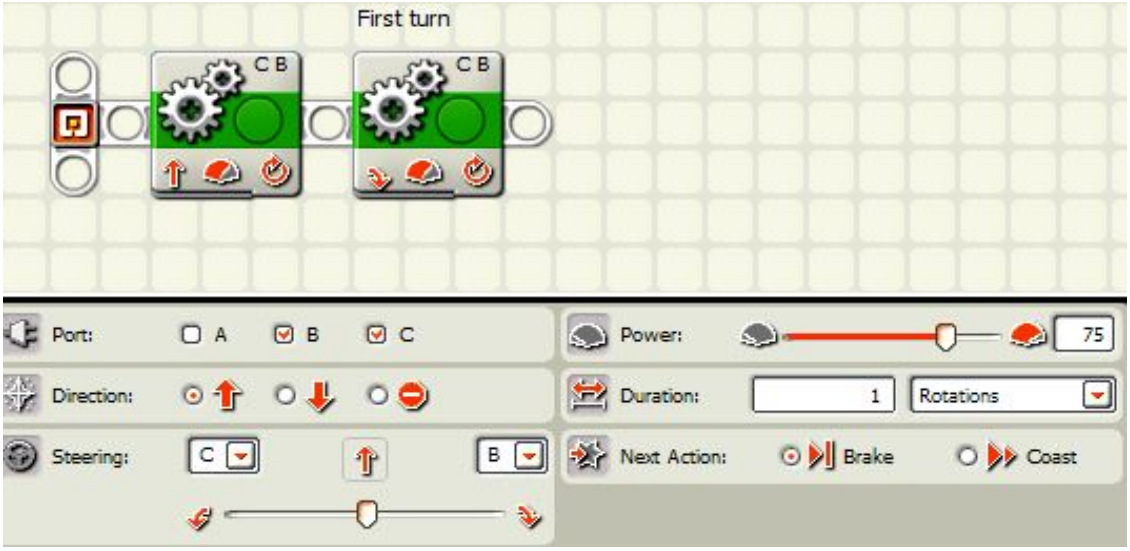
Sensors:
none

Directions:

1. Place a move block at the first of the program bar. Set the rotations to enough rotations to get it near the far side of the table. This will be about 3.5 feet. Refer to your engineers journal for the amount of rotations to go the distance.

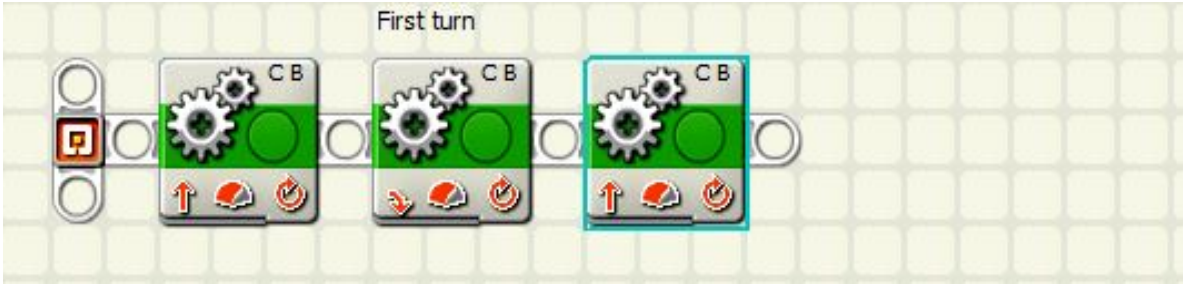


2. Place a move block on the line and slide the slider all the way tot the side so the robot will spin in place. Set the slider so the robot will turn right and set the rotations so the robot will turn 90 degrees.

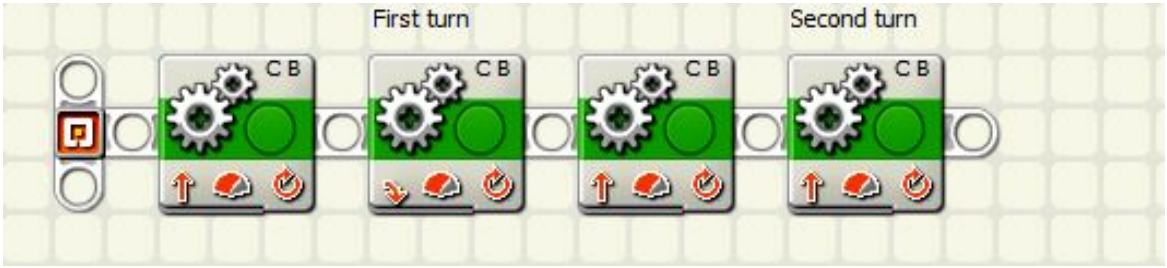


This is the first turn

3. Place a move block on the bar and set the rotations to get near the next wall. The number of rotations will be the same as step 1.



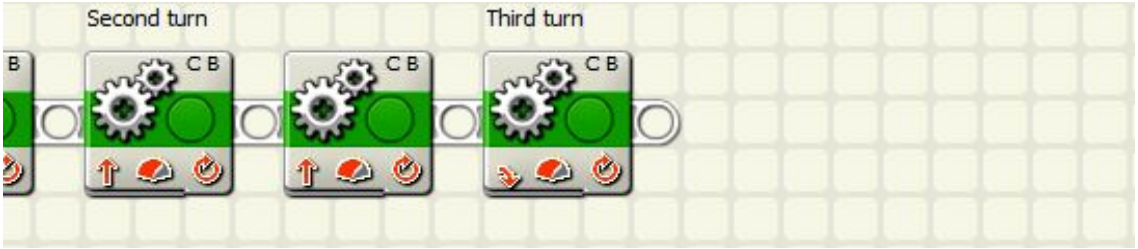
4. Place a move bar on the bar and set it the same as step 2 so the robot will turn to the right again.



This is the second turn.

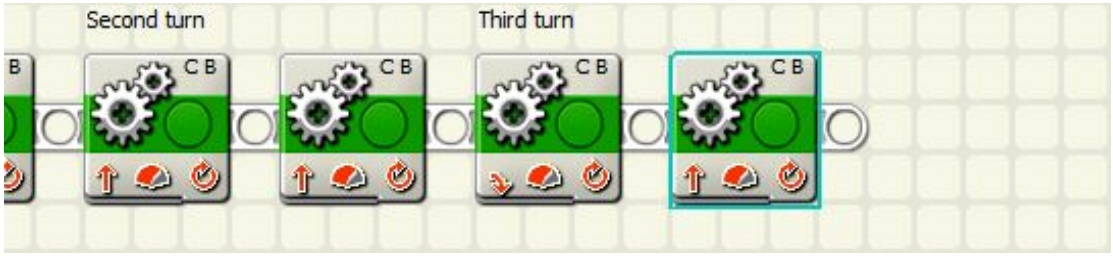
5. Place a move block on the bar and set it to go half way down the field. That would be half the rotations you used for step 2.

6. Place a move block on the bar and set it for another right turn like you did on step 2.



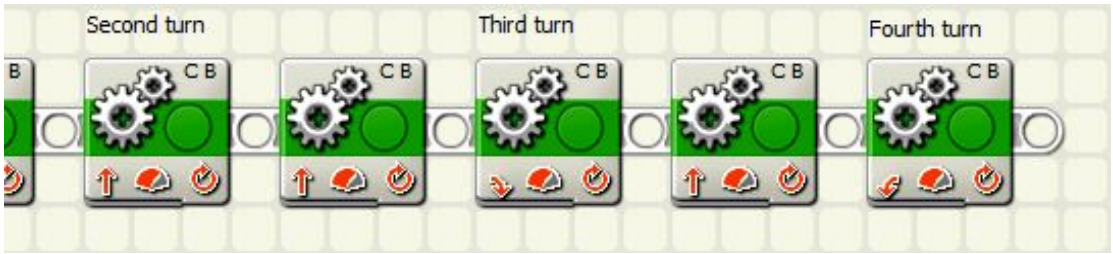
This is the third turn.

7. Place a move block on the program bar and set the rotations to the distance you did on step 3.



This step moves the robot between the two poles.

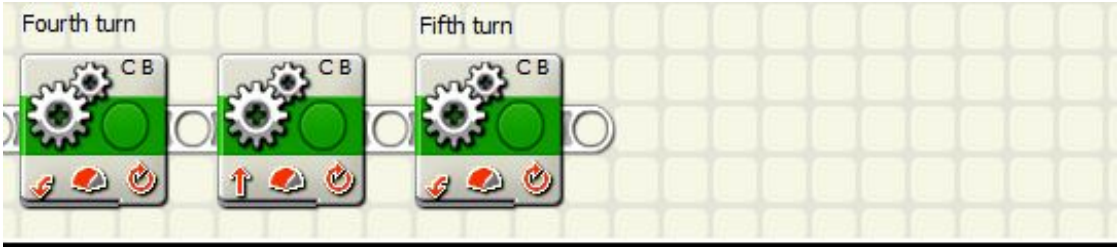
8. Place a move block on the bar and set it to turn like the other turns you did but this time it needs to turn the other way. It needs to turn to the left.



This step makes the robot turn towards the starting point.

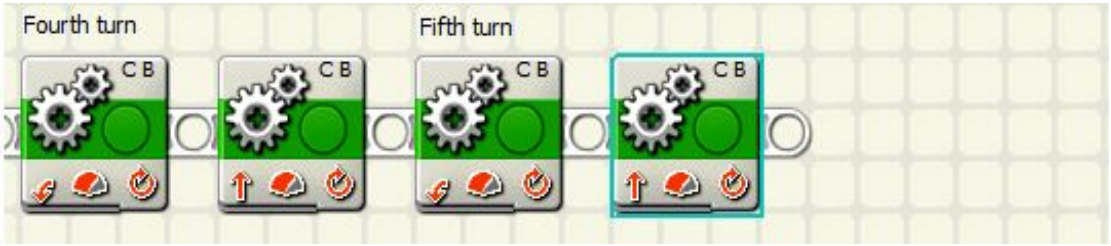
9. Place a move block on the bar and set the distance the same as you did for step 5.

10. Place a move block on the bar and set it to a left turn like you did in step 8.



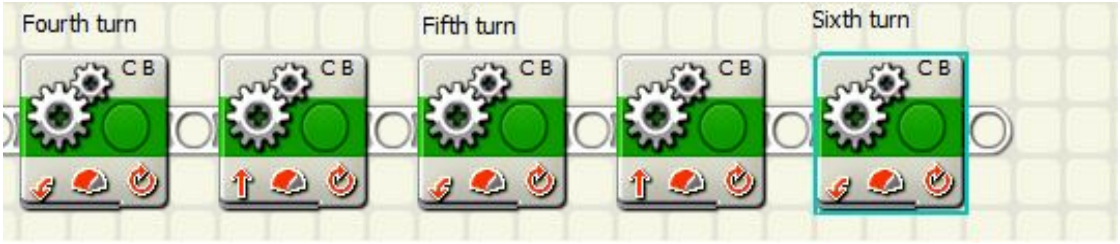
This should get the robot very near the starting point.

11. Place a move bar on the bar and set it to the distance you did for step 7.



This moves the robot along the bottom of the field and away from the starting point.

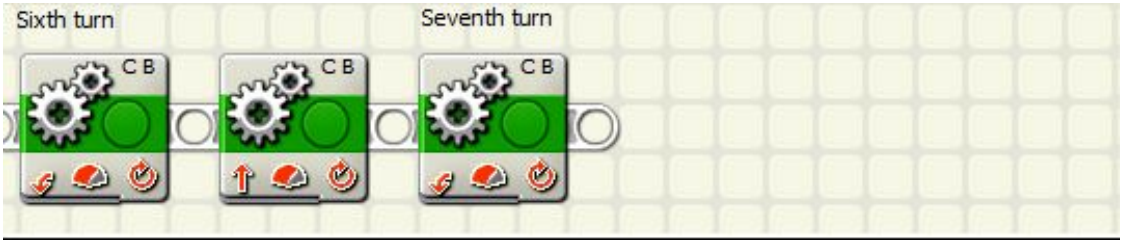
12. Place a move block on the bar and set it to a 90 degree left turn like you did on step 8.



13. Place a move block on the bar and set it to go half way up the field like you did on step 9.

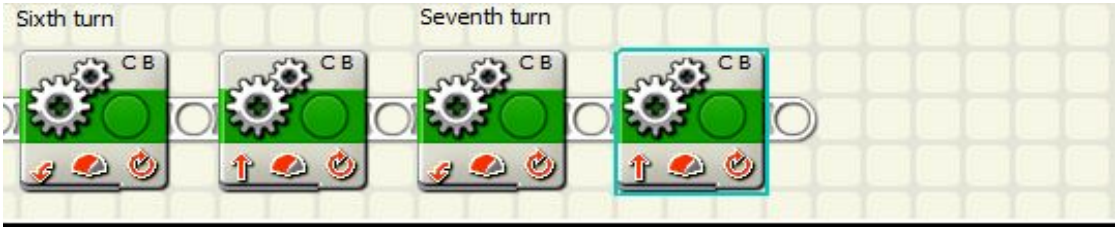
This moves the robot up the right side to the middle from top to bottom.

14. Place a move block on the bar and set it to a 90 degree left turn like you did on step 8.



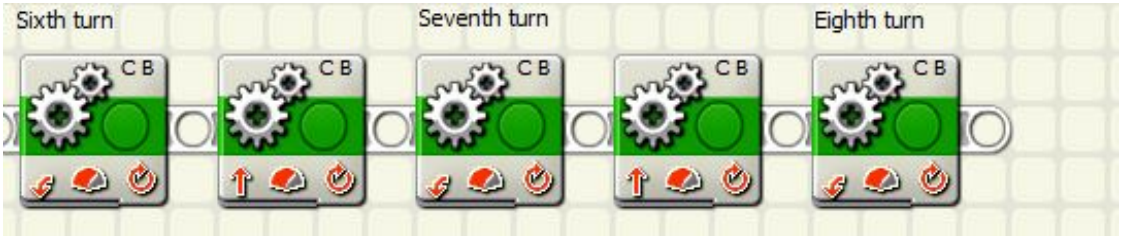
This should turn the robot so it is now facing between the two poles.

15. Place a move block on the bar and set it to the distance you set in step 11.



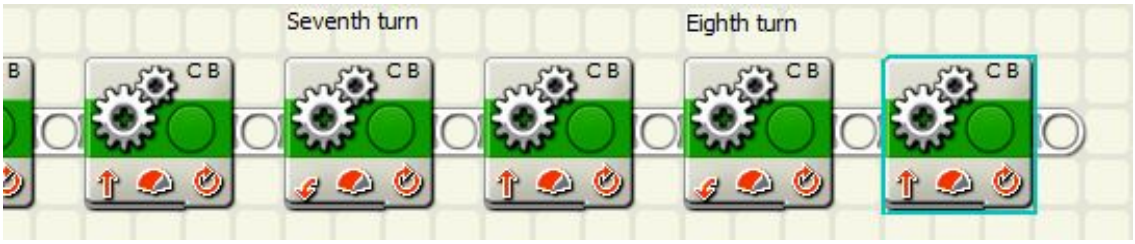
This moves the robot between the two posts and gets it just above the starting line.

16. Place a move block on the bar and set it to turn the robot 90 degrees to the left like you did in step 14.



This will turn the robot so it is now facing the starting point.

17. Place a move block on the bar and set the rotations to what you did on step 13.



This should get you to the starting point.

18. Make adjustments to all the straight move blocks and all the turns so that everything stays straight. It took a lot of blocks, didn't it? You can use loops to make it easier and keep your program shorter.

This can be a challenging mission. When programming this, the secret is to have the 90 degree turns as exact as you can. Tweak the various parts to keep the robot parallel to the walls.

One of the problems you may have is that the robot does not always turn exactly the same way each time, so sometimes your robot will be able to do the maze and other times it will run into a wall. Some ways to help the robot keep pointed in the right way is to use the **Shimmy Forward** and the **Shimmy Back** as explained later in the book.
Writing down the number of rotations for a 90 degree turn and the number of rotations for the robot to travel a foot and using these numbers to program your missions is a great way to save time. This way, there isn't so much trial and error so you can program faster.

Variations on Figure 8

The exercise can be changed to make it easier if you would like. One way to grade this is to give an A grade for doing the course as it is outlined doing the figure eight. Going around only the top bottle and returning back to the starting place can be graded a B, and going all the way around the outside in a big square can be a C grade.

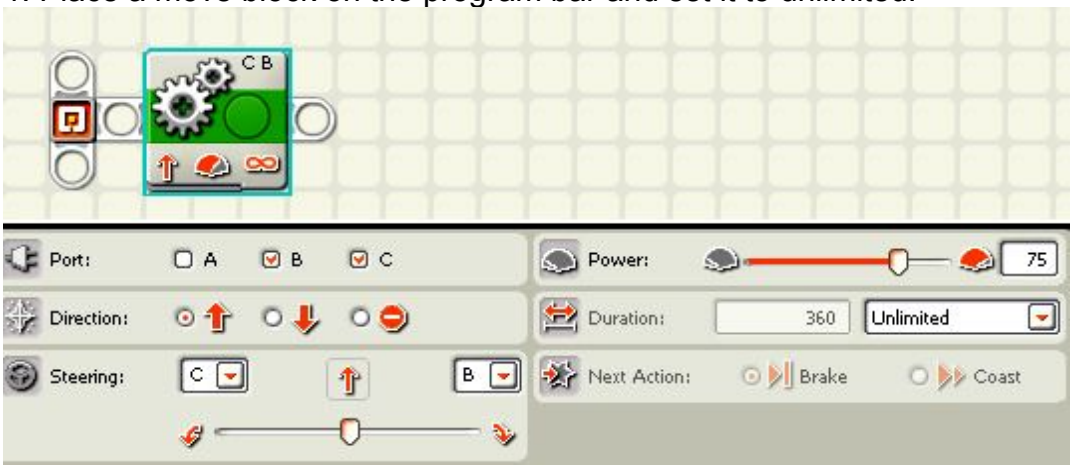
Figure 8 with Touch

Mission: see Figure 8, the mission before this one.

Equipment: Touch sensor

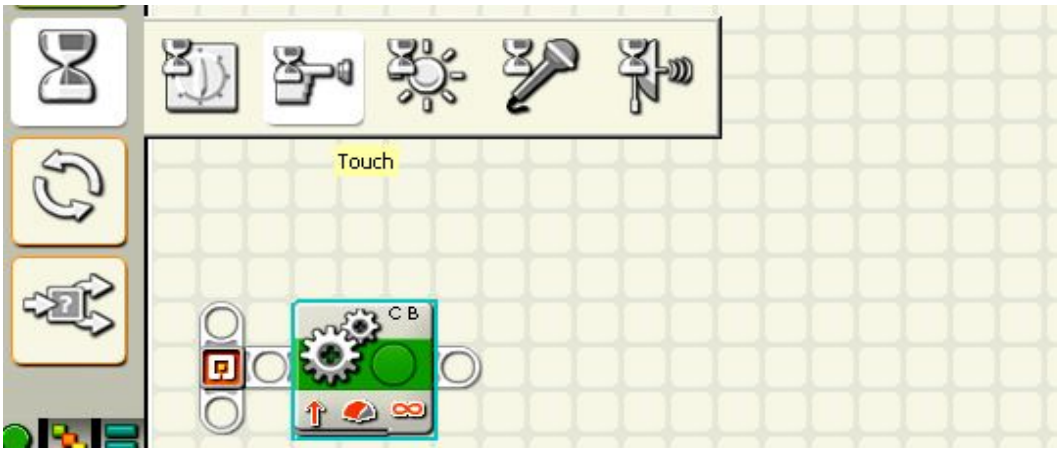
Directions:
These directions will not go through every step for this exercise. Instead, it will just show you the first turn and you will then know how to handle every turn after that.

1. Place a move block on the program bar and set it to unlimited.

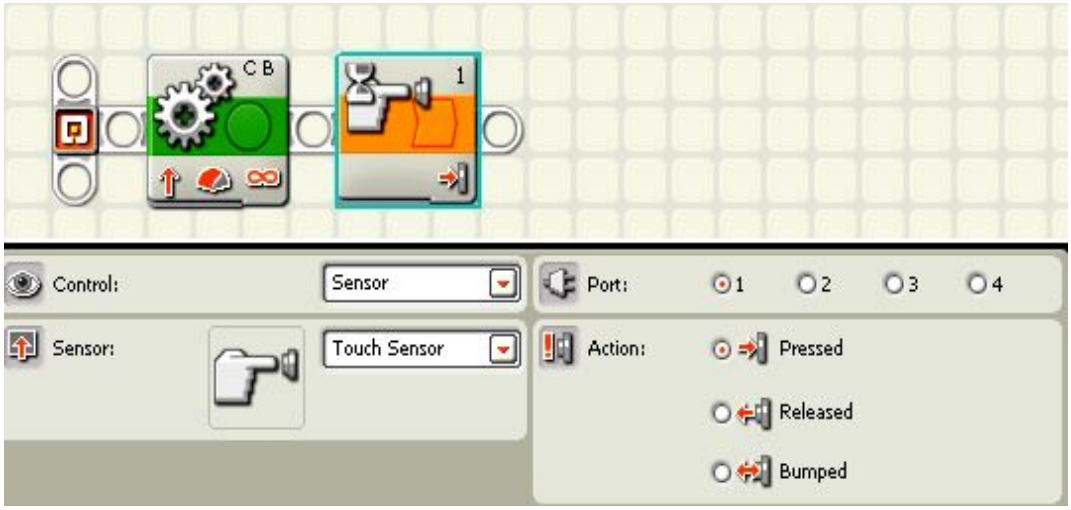


This will make the robot move for an unlimited distance until the next block tells the program to move on to the next block.

2. Place the cursor over the wait block and a menu will appear. Click on the touch block.

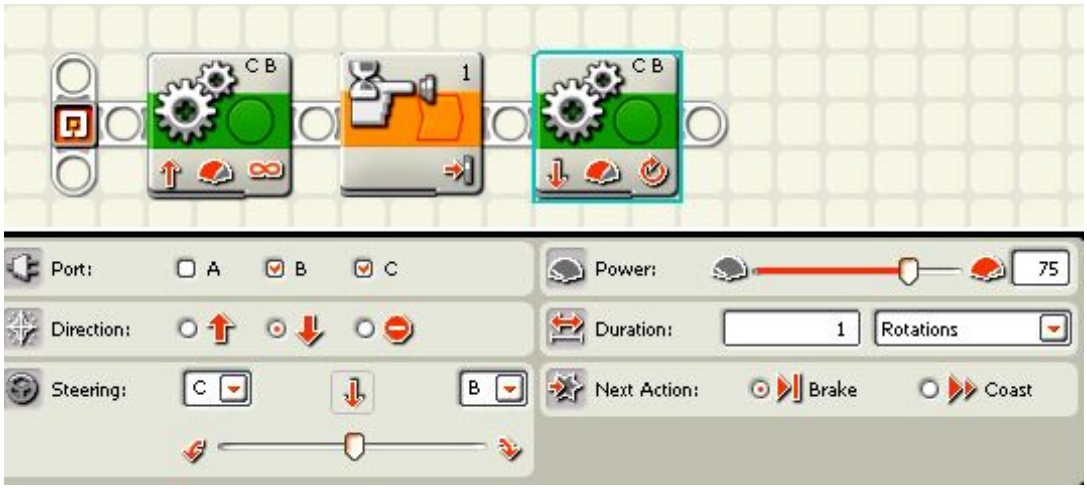


3. Place the touch block on the bar.



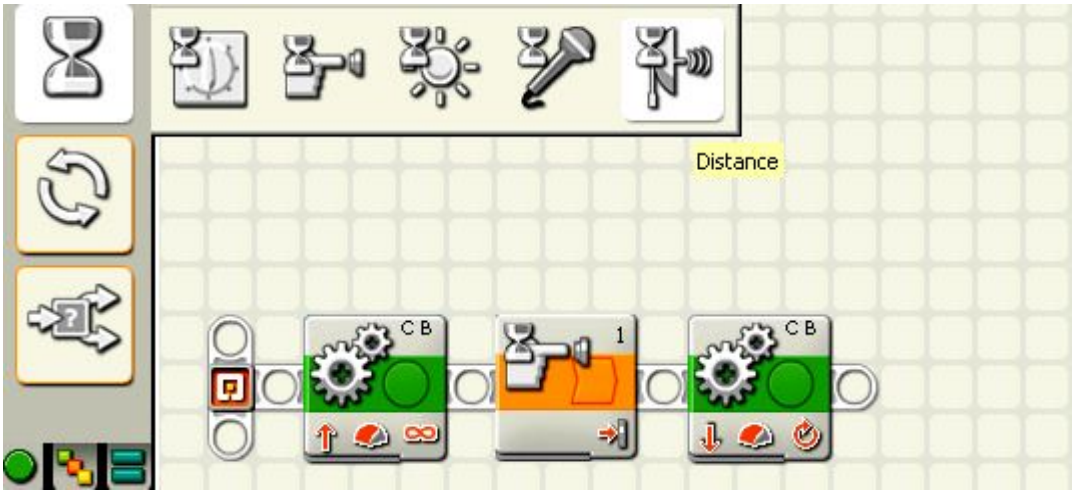
The touch sensor block will trigger the program to move to the next block when it is pressed.

4. Place a move block on the line and set it for reverse and leave it at one rotation.



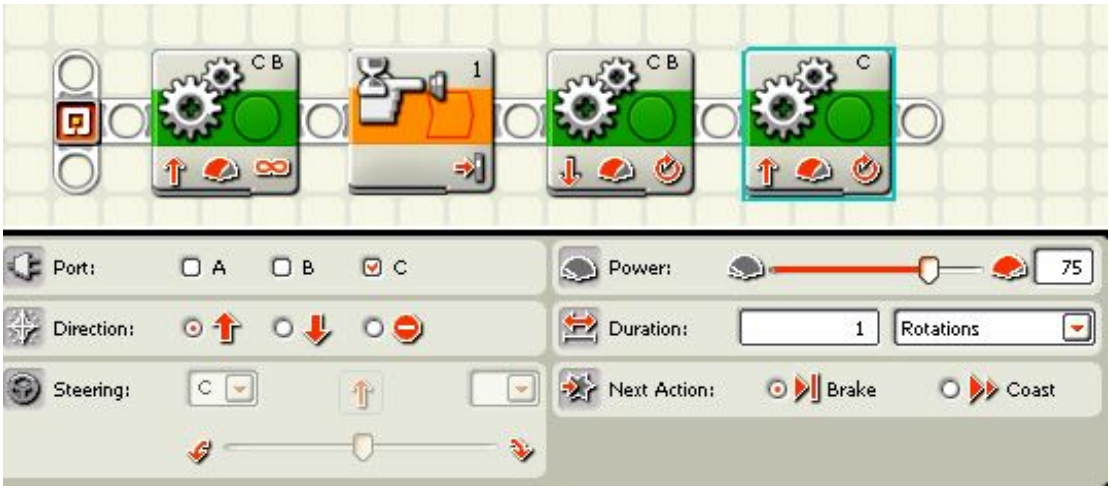
The rotations may need to be changed to make it move back enough that it will not hit when it turns, but close enough to wall to so it does not hit the post on the figure 8.

5. Place the cursor over the wait block and pick a distance block.



The distance block works with the ultrasonic sensor to measure distance by a ultrasonic ping. It works like a bat does with its sonar.

6. Place the move block on the line and click on the button next to Port B to remove the orange circle in it. Now only Port C has a circle in it.

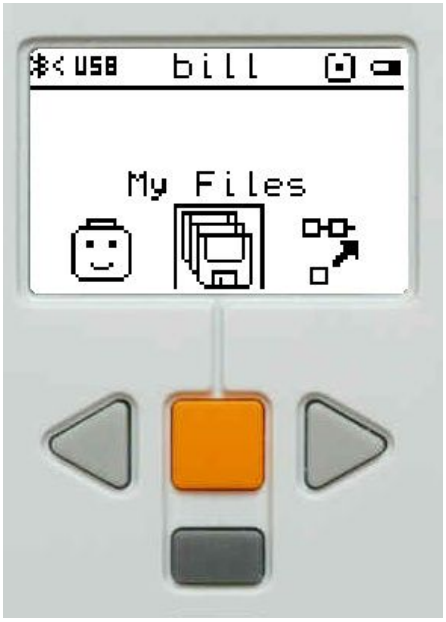
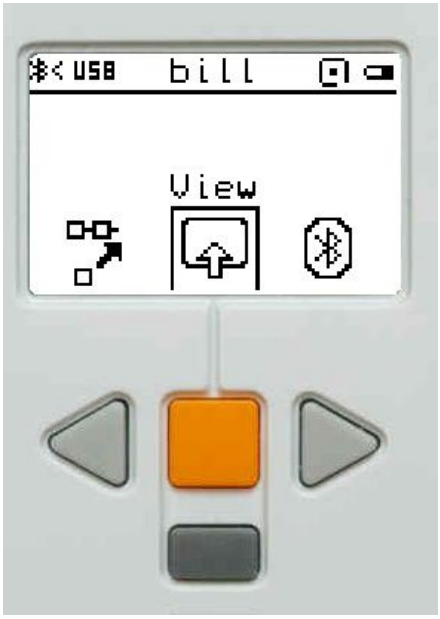


This means only the wheel attached to Port C will turn. This will make your robot turn to the right.

ULTRASONIC SENSOR

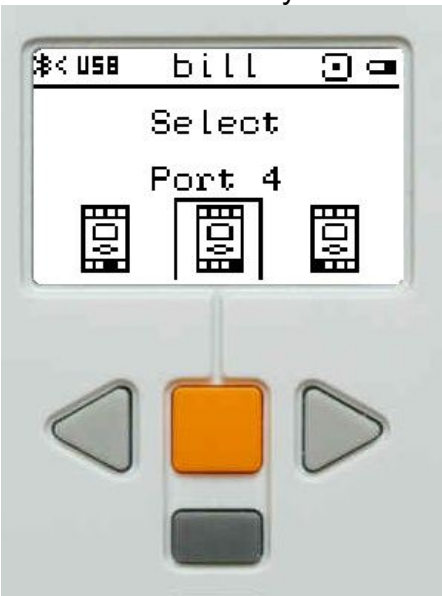
Readings

This is how you test a ultrasonic sensor inside the view section.

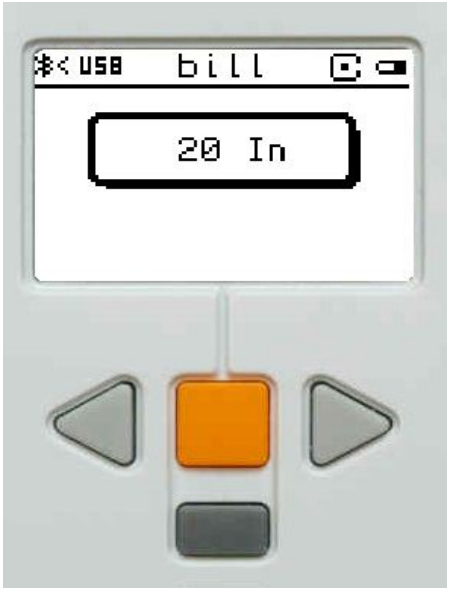
1. Start with the level that says My Files. 	2. Scroll to the right by pushing the right gray button until you see View. Push the center orange button.  *This will open up the view section so you can check the various sensors as well as the rotations of the motors.*

1. Push the right, gray button until you get to Ultrasonic inch (or Ultrasonic cm for centimeters). Push the center button.

2. Scroll sideways using the right gray button to get to the port you want. The ultrasonic sensor is usually uses Port 4.

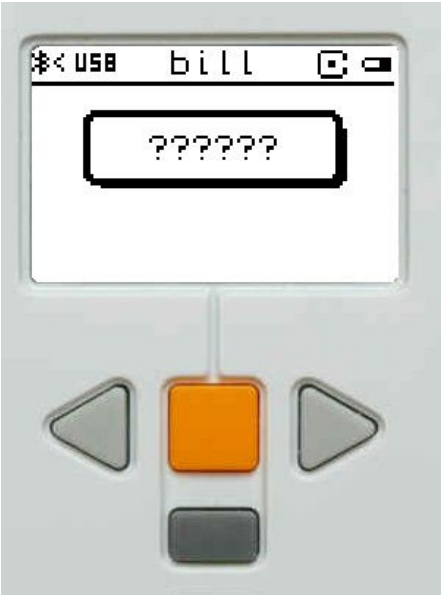


3. You can point the ultrasonic sensor at various objects and see what the sensor says the distance is.



Notice that the measurement is not very accurate. It will vary depending on the surface and it will not give a reading on a distance less than 3 inches in most cases.

4. If the sensor is not working or is not hooked up right, it will give you a series of question marks.



Check to see if the sensor is plugged into the same port as the one you are checking. Also, check to see the cord is plugged all the way into the brick and into the sensor.

Ultrasonic Readings

Mission:
The student will use the robot to test various objects to see how the ultrasonic sensor responds to various object depending on the angle, shape, and hardness of the object.

Equipment: ultrasonic sensor

Directions:
1. Attach the ultrasonic sensor so that it faces out away from the robot.
2. Set the robot so that the screen shows the ultrasonic sensor in inches or centimeters and set it one foot away from the first object.
3. Write down the reading of the first object in on your hand out. Move on to the next object and get a reading for that. Continue to do the same for each object.

	Situation	Reading in inches or centimeters
1	wood straight	
2	wood 45 degree	
3	2 liter pop bottle	
4	towel on wood	

Explain why the readings were different in your opinion. How would the various surfaces make a difference to the readings.		
1	wood straight	
2	wood 45 degree angle	
3	2 liter pop bottle	
4	towel on wood	

Turn this in to the teacher.

Figure 8 with Ultrasonic

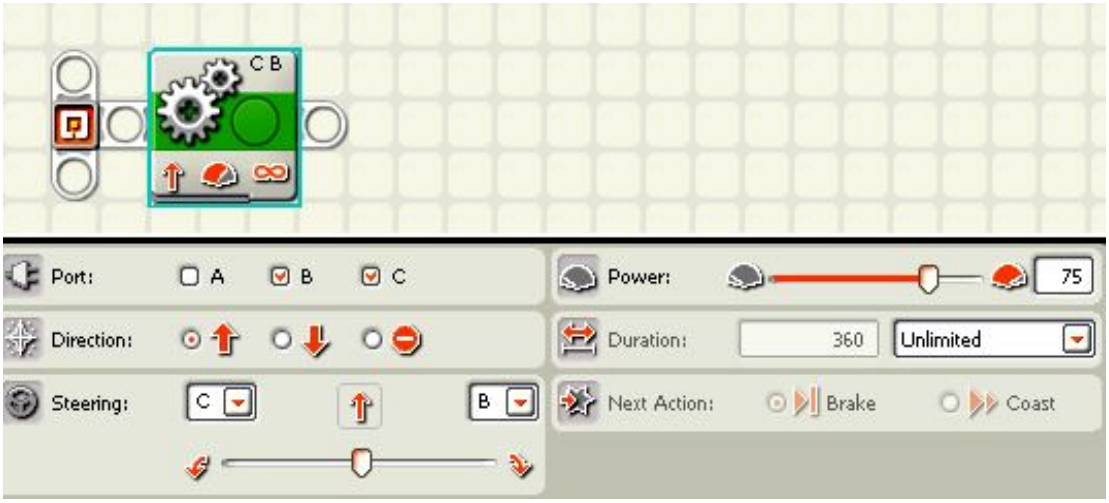
Mission:
See **Figure 8**, two missions before this one.

Equipment:
Ultrasonic sensor

Directions:

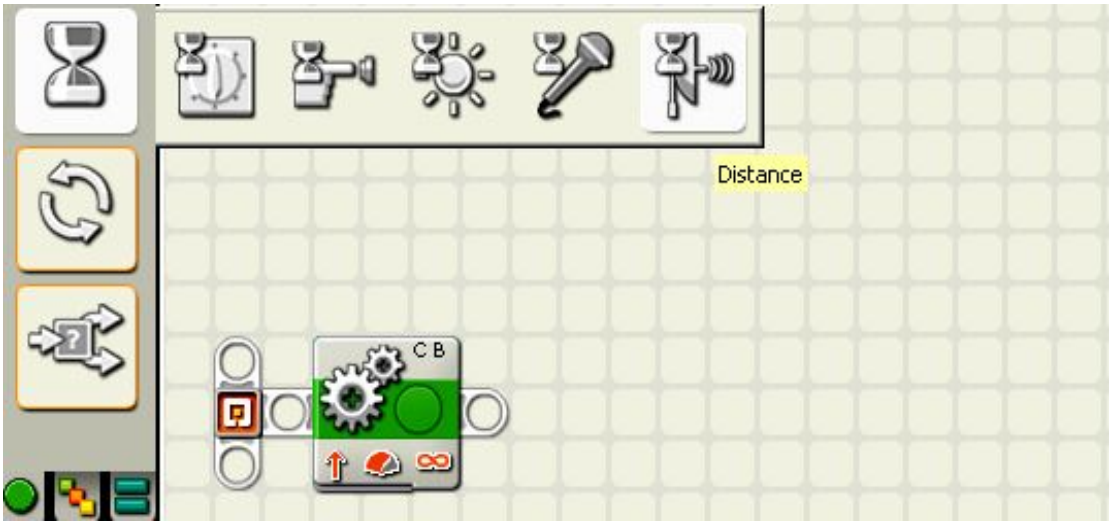
These directions will not go through every step for this exercise. Instead, it will just show you the first turn and you will then know how to handle every turn after that.

1. Place a move block on the program bar and set it to unlimited.

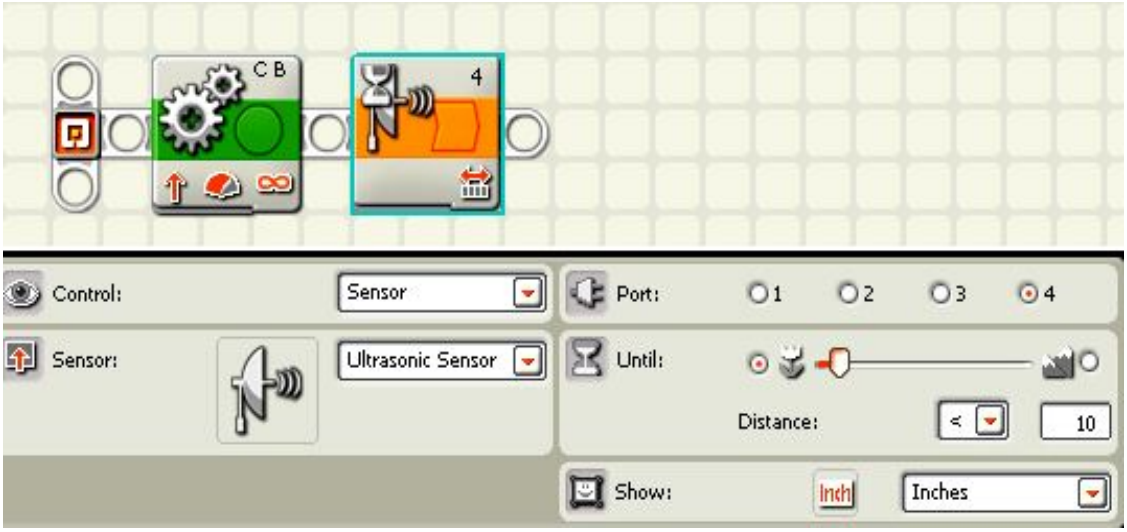


This will make the robot move for an unlimited distance until the next block tells the program to move on to the next block.

2. Place the cursor over the wait block and a menu will appear. Click on the distance block.

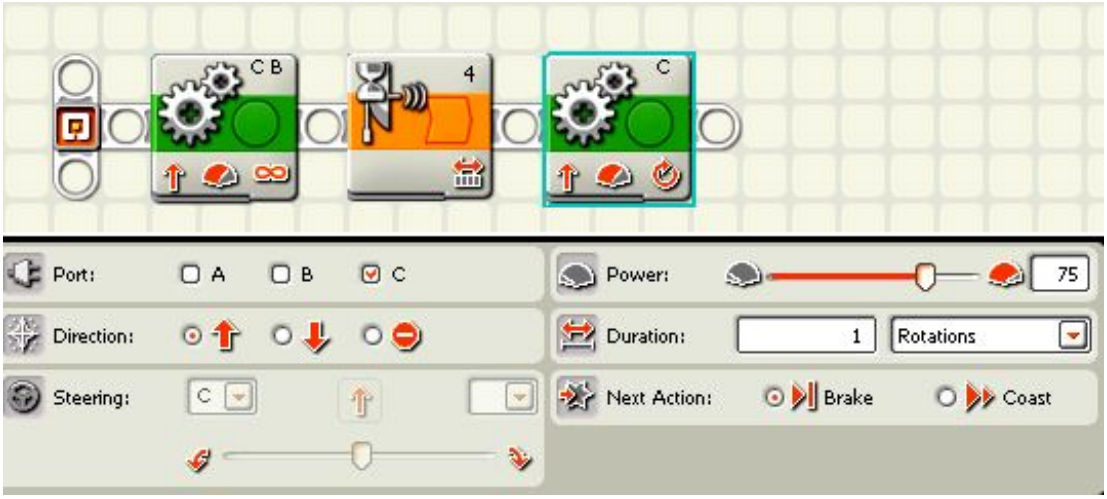


3. Place the distance block on the bar.



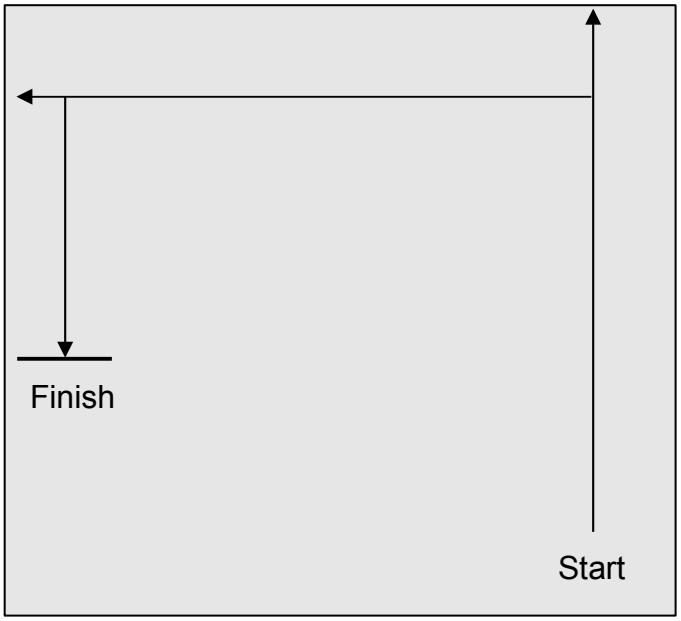
The distance sensor block will trigger the program to move to the next block when it is pressed.

4. Place a move block on the line and set it for reverse and leave it at one rotation.



The rotations may need to be changed to make it move back enough that it will not hit when it turns, but close enough to the wall so it does not hit the post on the figure 8.

2 Bumps and Stop



Mission: The robot will bump into one wall, turn left, bump into the next wall, and then stop when it senses a dark line under it.

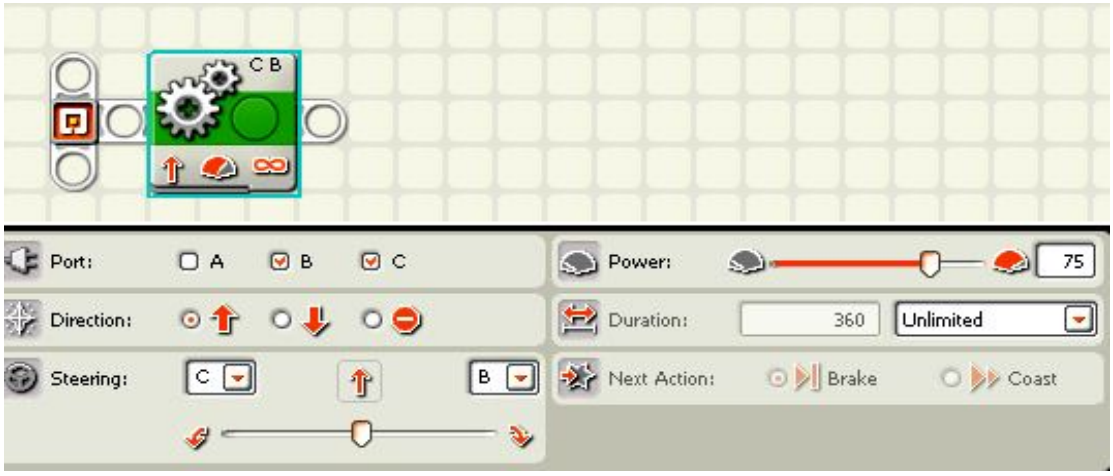
Equipment: White table top or playing field.
Blue or green masking tape to make a line

Sensors: Touch and Light

Directions: Attach the light sensor facing down and the bump sensor facing forward.
Adjust the light sensor so it is at least two pennies above the surface.

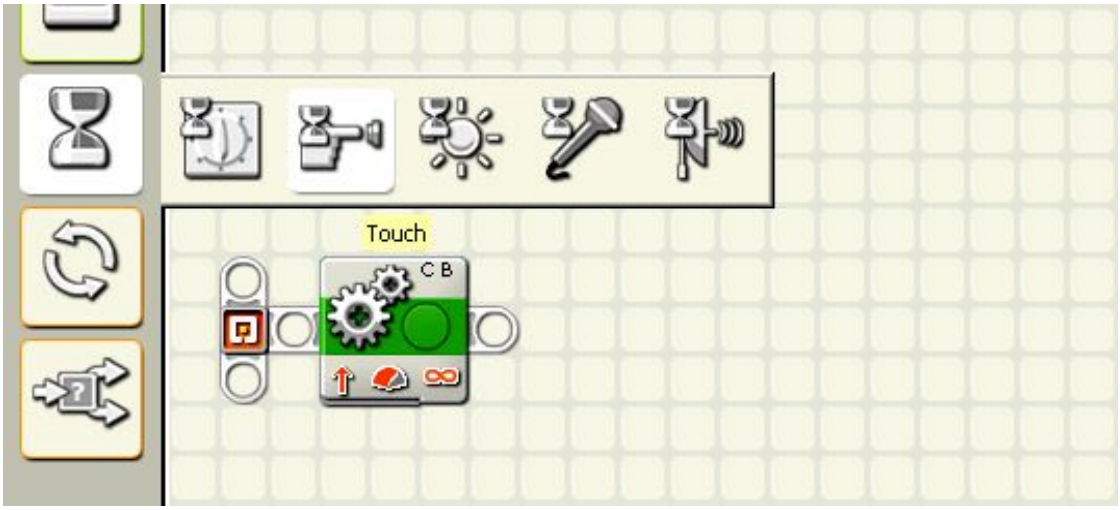
Light sensor reading on white	Light sensor reading on tape

1. Place a move block and set it to unlimited.

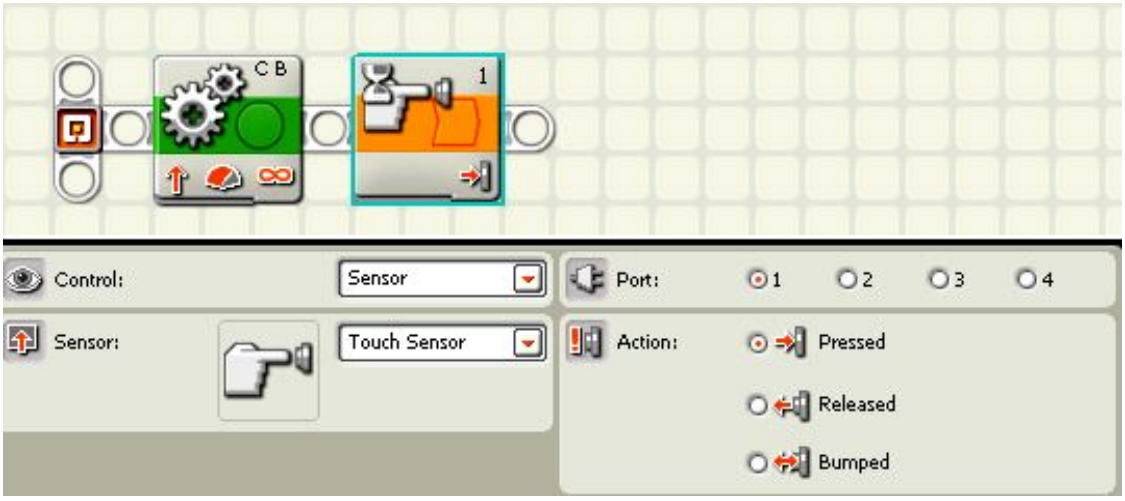


This makes the robot go forward until it is told to do something else by a wait block.

2. Place your cursor over the wait block symbol and then click on the touch block. It looks like a finger touching a button.

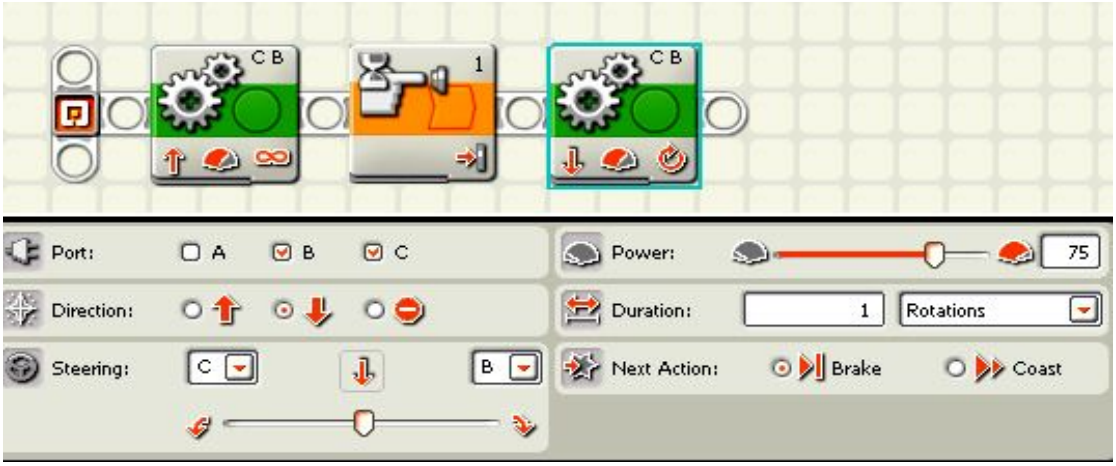


3. Place the wait block. Notice it is set to port 1 (see it on the middle right side of the illustration). You can change it or leave it as it is.



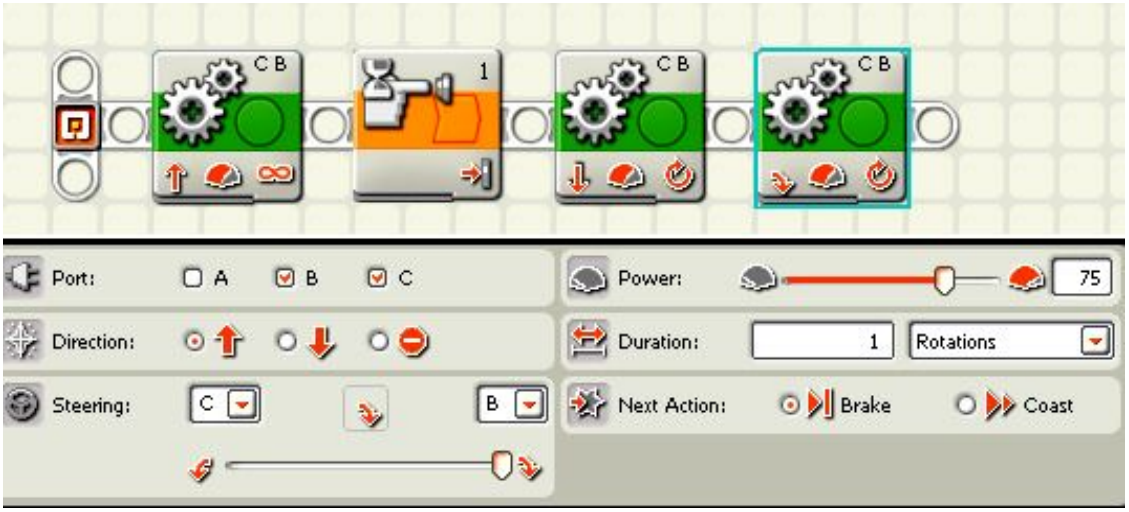
The touch block will stop the unlimited rotations of the motors from the move block when something presses the touch sensor for longer than 1 second. The motors will keep turning until the 1 second is up.

4. Place a move block and set it to reverse and leave it at one rotation.



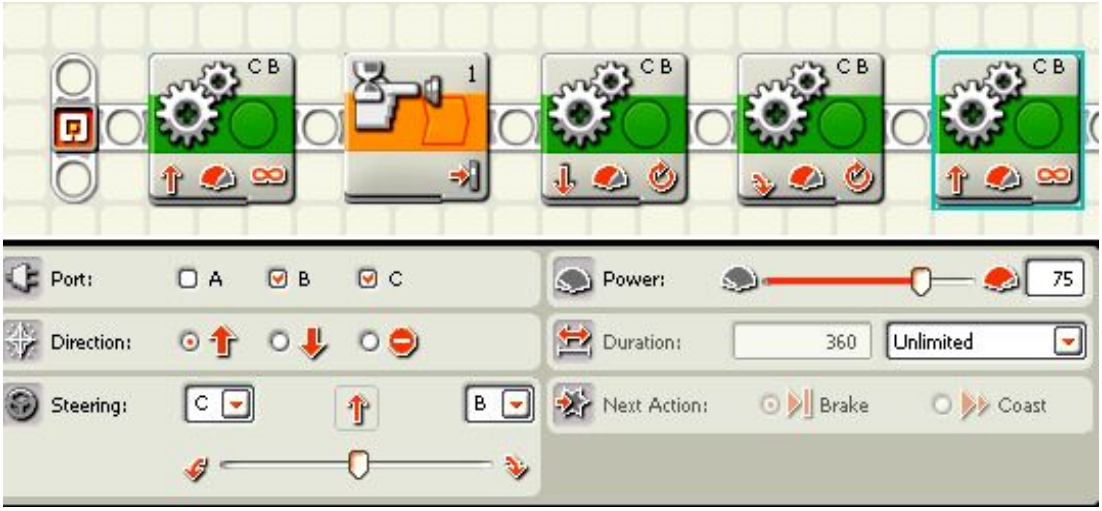
This will make the robot back up so that when it turns, it will not hit anything and mess up the accuracy of the turn.

5. Place a move block and slide the steering bar all the way to one side or the other.



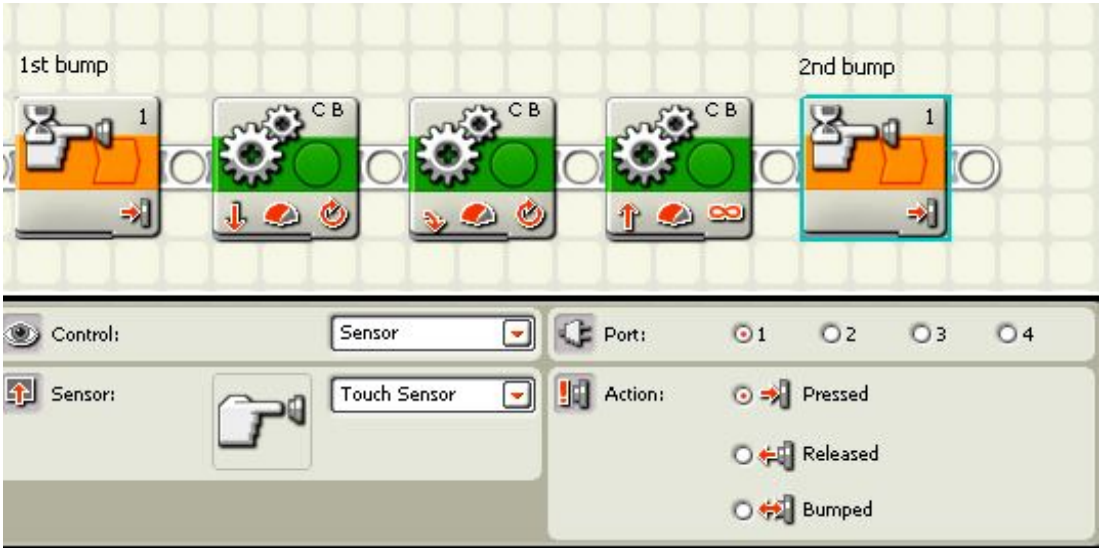
This makes the robot turn in one place by making one wheel turn forwards and the other wheel turn backwards.

6. Place a move block on the program bar and set it to unlimited.



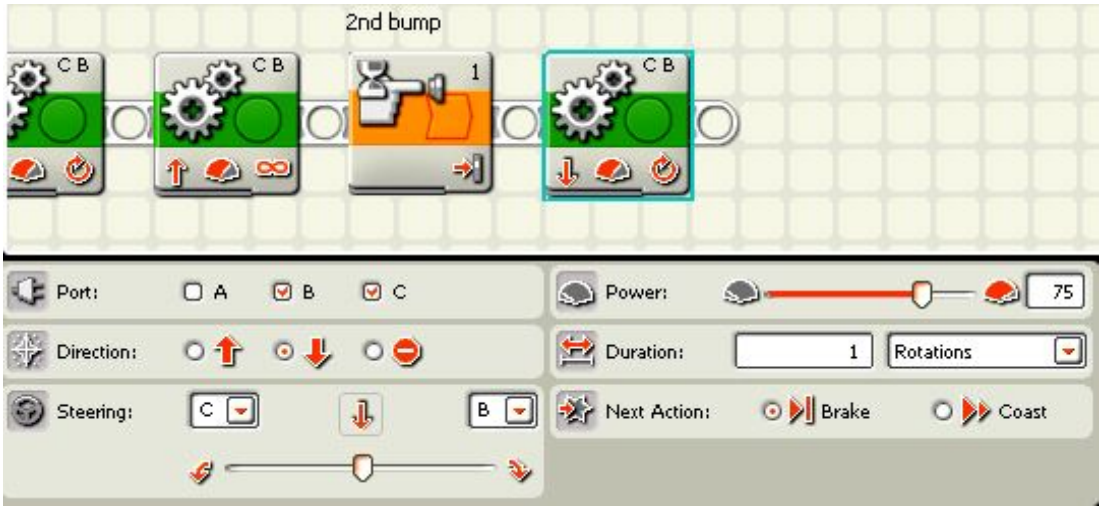
This makes the robot go forward until it is told to do something else by a wait block.

7. Place another touch block and leave it set to port 1.



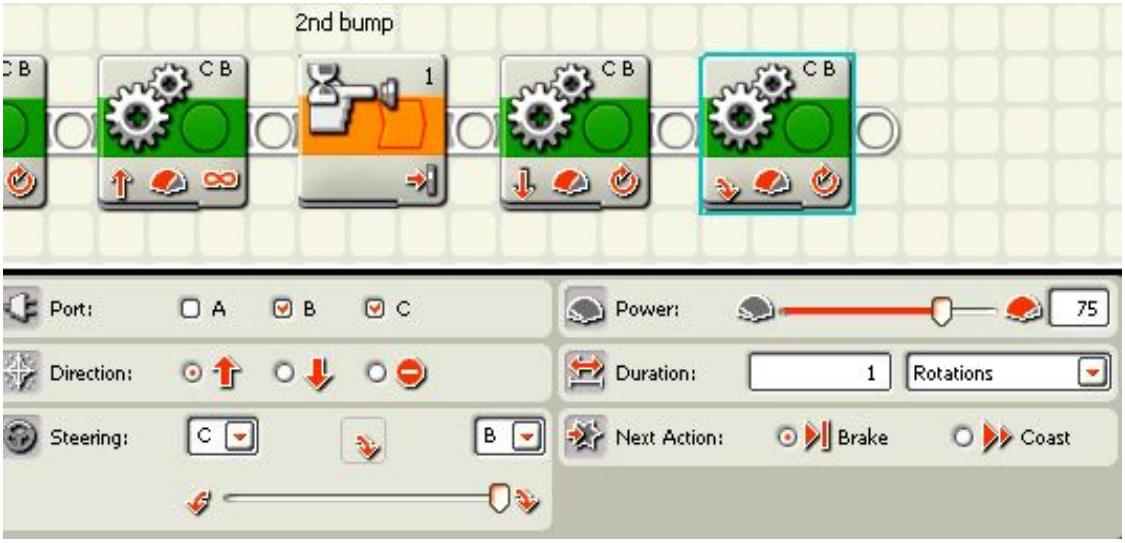
It is better to set up the sensors so they use the default ports whenever possible so that you will never accidentally forget to change one of the ports and wonder why your program is not going correctly.

8. Place a move block on the line and set it to go in reverse for one rotation.



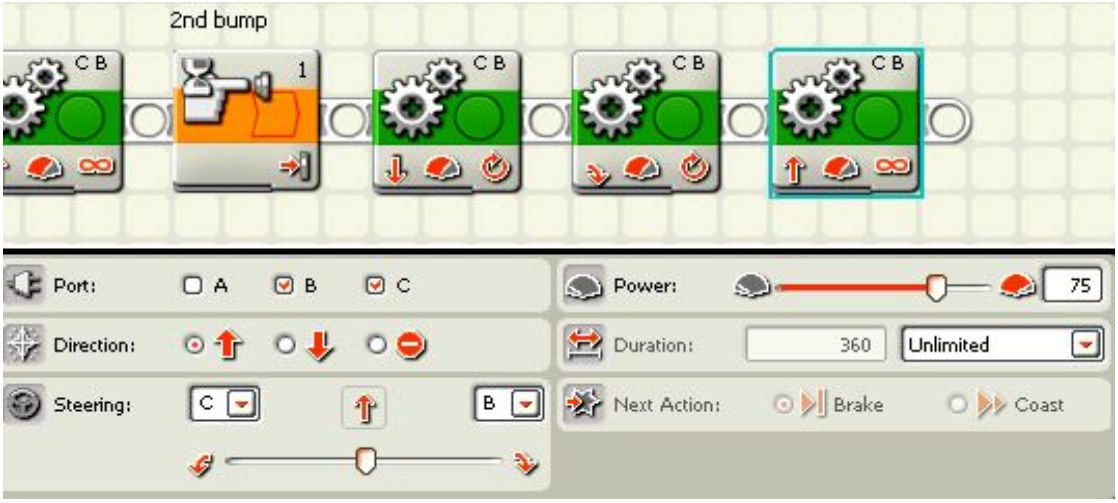
This is again to make the robot back up so that it will not hit the wall when it is turning.

9. Place a move block on the line and set the steering all the way to the side like you did in step 5, Set it to the same side as you did before.



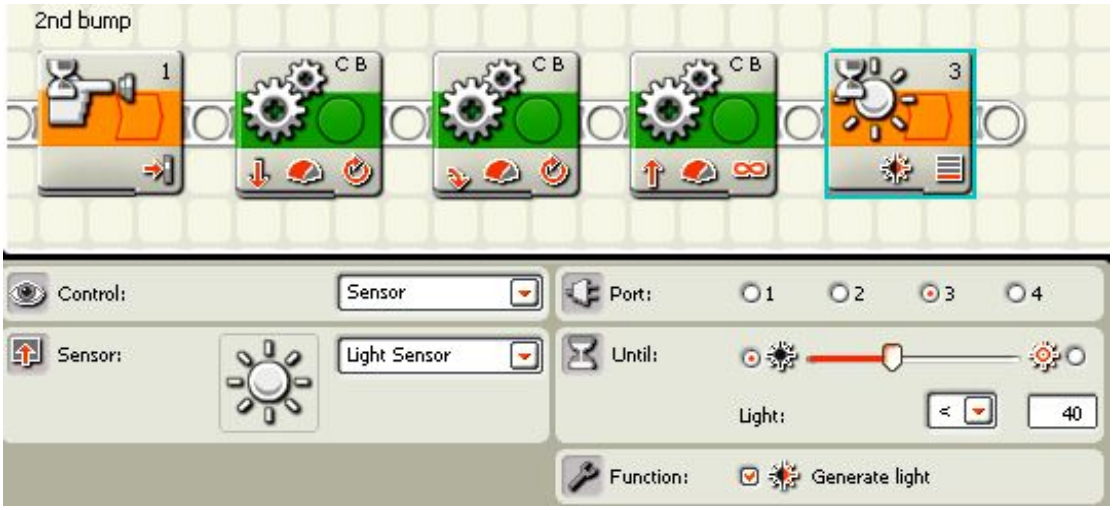
This makes the robot turn.

10. Place a move block and set it to unlimited again.



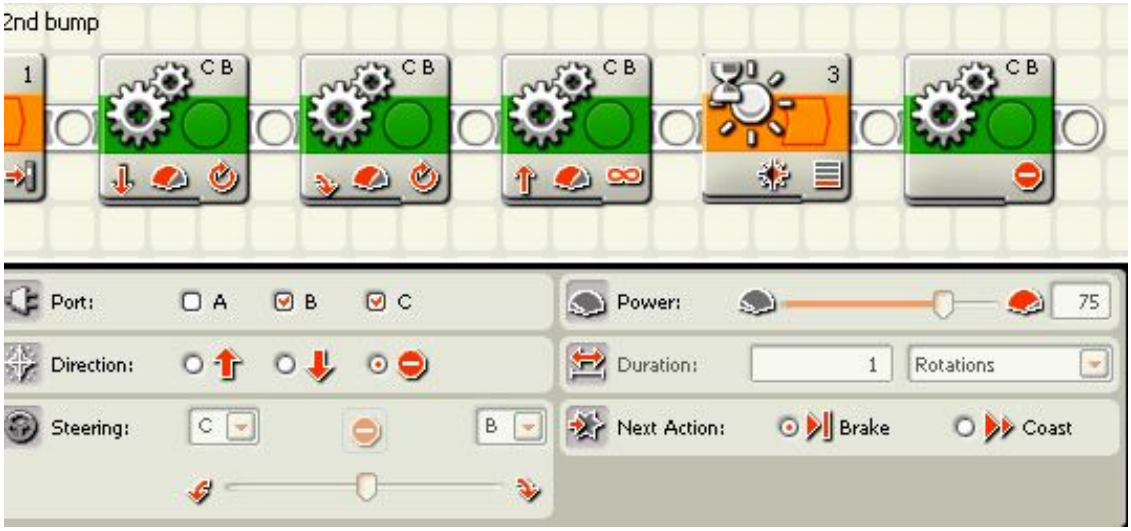
This makes the robot keep going forward until the next wait block tells the robot to move on to the next block in the program.

11. Put the cursor over the wait block and pick a light sensor. As you can see, it looks like a sun shining. Place it on the program bar. Set it to less than and 40 or so.



It is best to set the light sensor to about 5 points less than the reading on the white paper. That gives the sensor a little leeway on any differences of the lighting so it won't accidentally trigger the next step when you don't want it to be triggered.

12. Place a move block on the line and set it to stop. You do that by clicking the little radio button on the lower left side by the reverse button.



This block stops the turning of the wheels. It is like putting on the brakes of a car or bike.

Wait Block

An Explanation

Wait blocks are used when you want an action to keep going until something happens. It is like when you stand at your bus stop waiting for your bus to come. The bus comes and then you get on the bus. So we would write this out like this:

How long do you stand there? It could be a few seconds or it may be several minutes. You don't know exactly. You just know that you will continue to stand there until the bus arrives and that will be your signal for you to go on to the next step which will be to get on the bus.

Here is a short explanation of the different types of wait blocks:

	**Time:** You can set the number of seconds to wait before the next program block starts.
	**Touch:** You can set it to wait until it touches something before the next program block starts.
	**Light:** You can set it to wait until it senses something light or dark before the next program block starts.
	**Sound:** You can set it to wait until it senses a sound before the next program block starts.
	**Distance:** You can set it to wait until it senses an object by saying if it is closer or farther away than a set number of inches or centimeters before the next program block starts. Using the ultrasonic sensor, it bounces an ultrasonic beep off of an object and measures the distance. It works like how a bat uses sound to find distance. It's like sonar.

Light Sensor,

An Explanation

When you first put a light block on the program line, the light block is set with a greater than sign and the threshold of 50. Than means that the light block will be triggered by any amount of light above the 50 percent level. This is not very consistent since the distance of the surface and the brightness of the surrounding light will influence it. It is best to take measurements of the light sensor setting the robot down on the field with the light sensor over the white surface and then seeing what the percentage is over the dark line. Write down both percentages so that you know what levels you need to use. Also, the exercises call for the sensor being triggered by a dark line so in almost every mission, we will change the greater than sign to a less than sign. This means the light wait block will trigger when the it senses something darker than the set percentage—like a dark line, for instance.

Attach the light sensor so that it faces down. It needs to be at least ¼ of an inch off the floor. It can be higher but sometimes it isn't as accurate at reading the reflected light if it is.

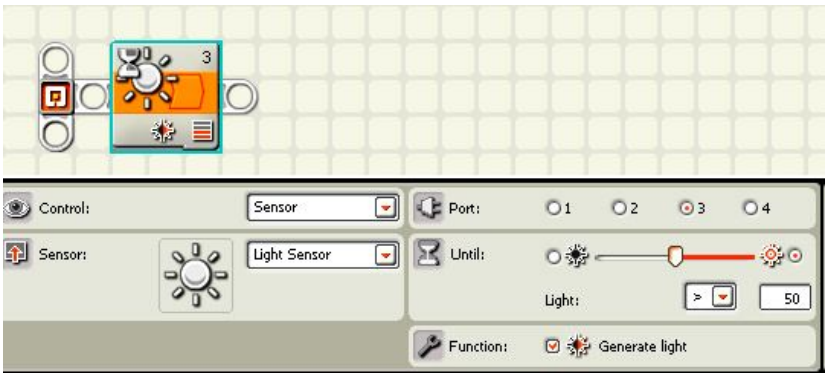
Don't make the robot move too fast. Sometimes the robot will move right past the line before the program can react and make the robot back up and turn.

Program the robot so that it will move forward until it senses the line of tape; then, have it back up, turn, and continue on moving forward again. Use a loop so that it will continue indefinitely.

Light Block,

An Explanation

This is the light sensor block and the screen that is below it. Notice that on the block is a number 3. This means that the light block will be looking for a reading from Port 3. If the light sensor is not plugged into a different port, it will not get its readings. You can change what port you want the block to look into, but it is better not to since you might remember to change it once or twice, but forget to change it on some of the light blocks in other parts of your programs, so just leave it in Port 3 unless you have a real good reason not to plug it there.



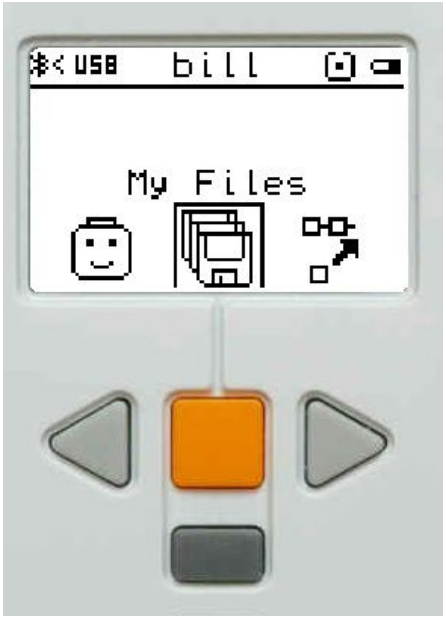
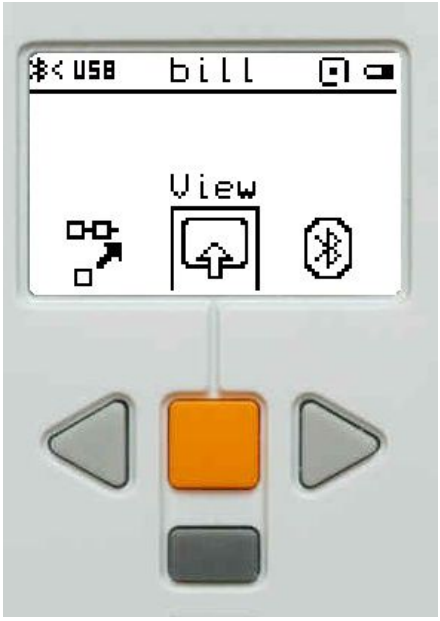
Notice that you can choose a greater than symbol “>” or a less than symbol “<” on the right side. Most of the time you will want to find a dark line, so you will want to look for something whose brightness is less than the surrounding surface, so you will usually select the less than symbol “<.”

On the bottom right side, there is a radio button that says “Generate light.” This will default to this setting. It means that the light sensor will shine a red light. This is what you usually want since you want the robot to shine a light on the surface and sense how dark or light the surface is so that the robot can stop on a line or follow a line. Sometimes you may want to measure the light in the room or under a shade of some kind. In this case, you want to click on it so that the light does not shine.

You can adjust the light level two ways. You can use the slider or you can delete the numbers in the box and type in new numbers. Either way works just fine. It is just what you are comfortable doing.

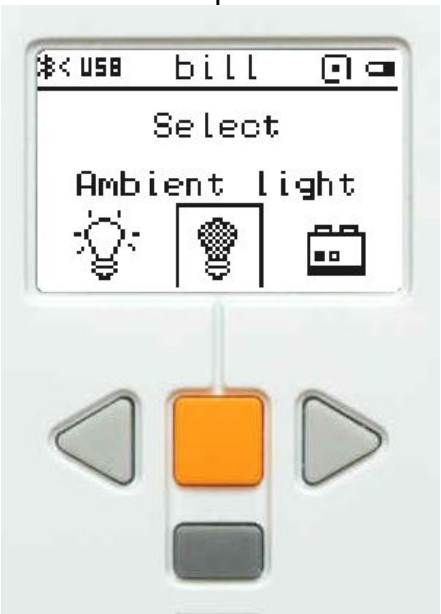
LIGHT SENSOR READINGS

This is how you can check a light sensor inside the view section.

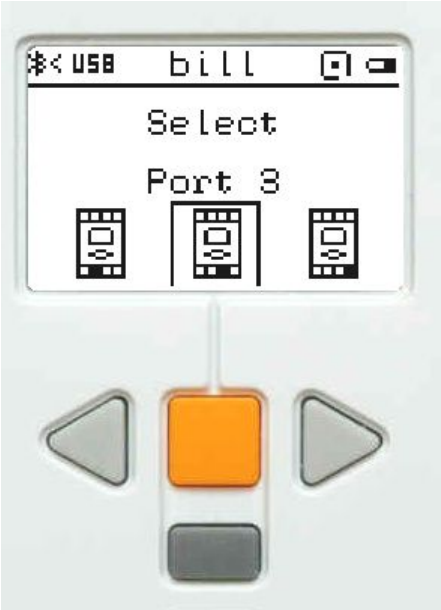
1. Start with the level that says My Files. 	2. Scroll to the right by pushing the right gray button until you see View. Push the center orange button.  *This will open up the view section so you can check the various sensors as well as the rotations of the motors.*

1. There is two ways to use the light sensor. One is to have the light sensor shine its light and have the electric eye measure how much light bounces back. Press the center orange button to access this part.

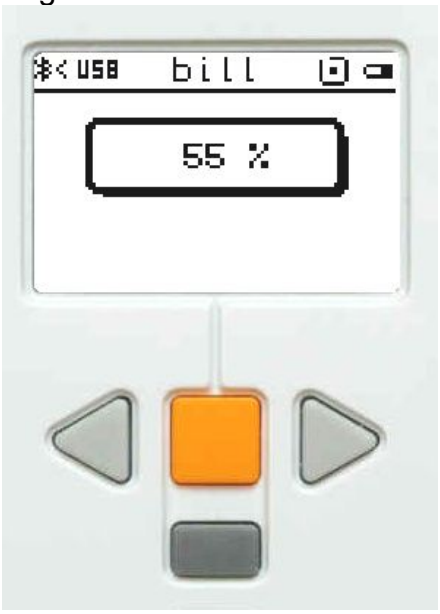
2. The other way to use the light sensor is to not shine the light and just measure the light in the room. The dark light bulb means the little red light on the sensor does not turn on. Press the center orange button to access this part.



3. The light sensor is usually attached to Port 3.



4. The light sensor will show as a percentage on the screen.



Light Readings

Mission: The student will use the robot to see what readings the light sensor will give on various objects and in various lighting conditions.

Equipment: light sensor

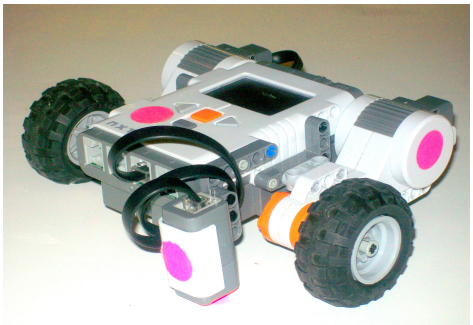
Directions:

1. Attach the light sensor so that it faces down to the surface below the robot.
2. Set the robot so that the screen shows the light sensor shining.
3. Scroll over to Port 3 which is the default port of the light sensor.
4. Set the robot on the first object.
5. Write down the reading of the first object in on your hand out. Move on to the next object and get a reading for that. Continue to do the same for each object.

Generating light	
Surface	**Reading**

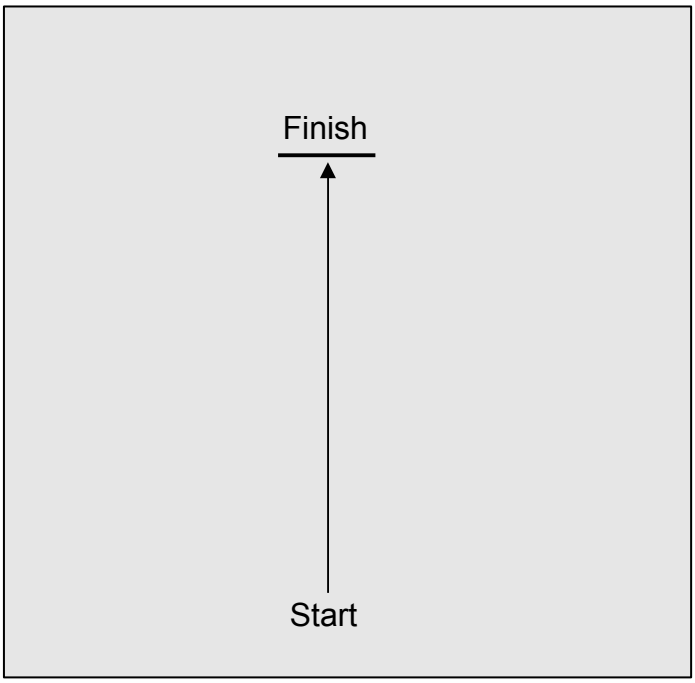
Not generating light	
Surface	**Reading**

Set the light sensor so that it faces down and that it is about ¼ to ½ inches from the surface so the light can
hit the surface and bounce back to the light sensor.



Stop on the Line

Mission: the robot will move forward indefinitely until the light sensor senses a black line, then the robot will stop.

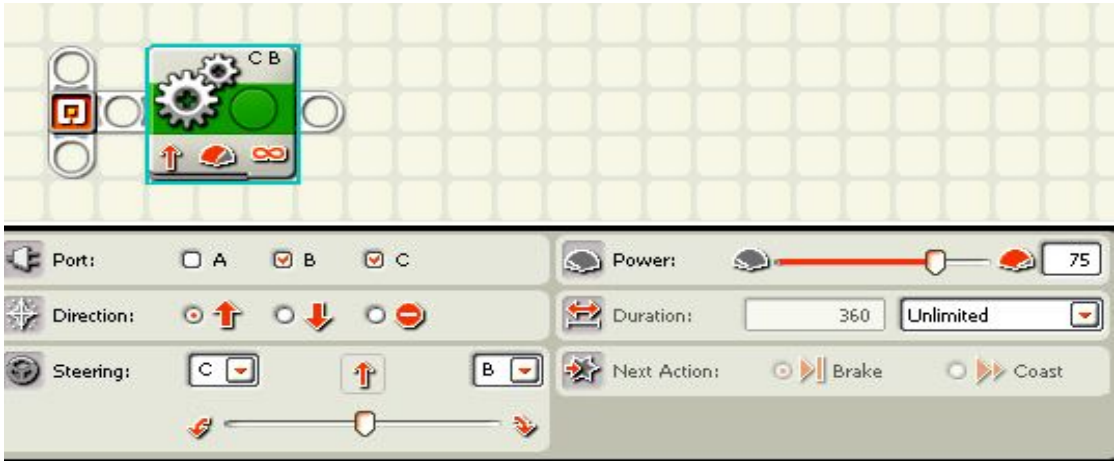


Equipment:
White table top or playing field.
A line of tape from narrow side of the table to the other narrow side

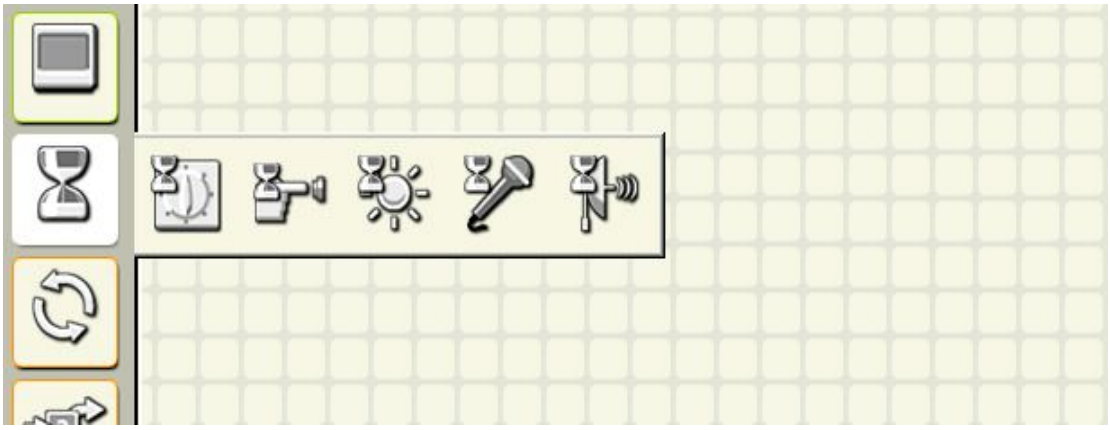
Sensors:
Light

Directions:
Attach the light sensor so that it faces down.

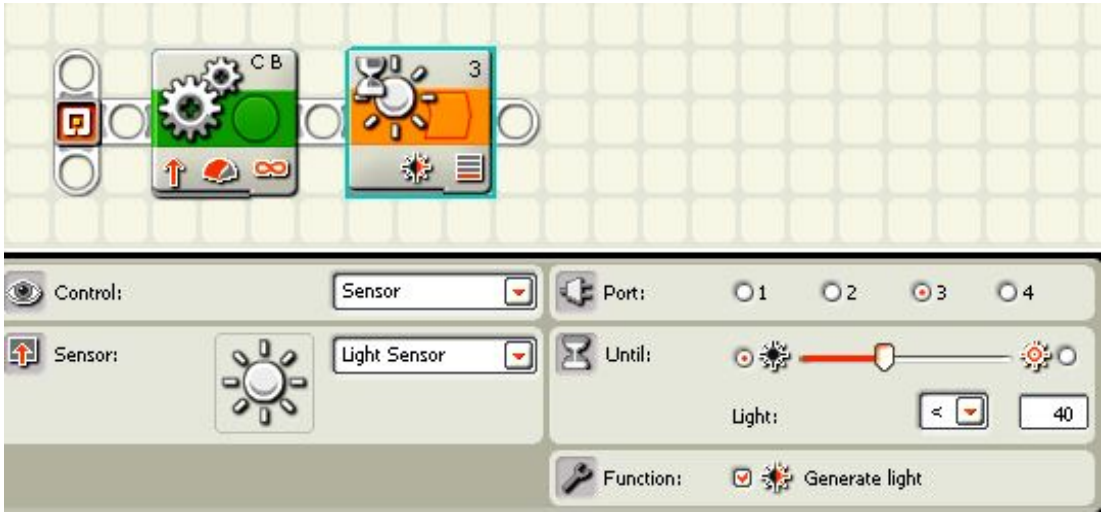
1. Place Move block at the program start.



2. Move cursor over the wait block and click on the Light wait block that looks like a sun. That is a light wait block.

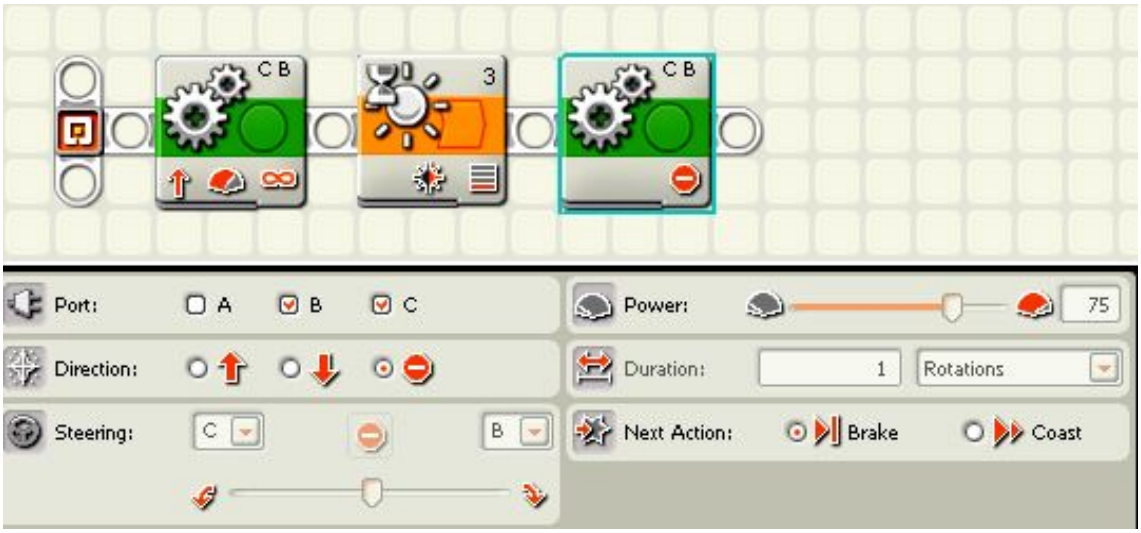


3. Place a light wait block on the line.



This tells the program to stop moving and do the next part of the program.

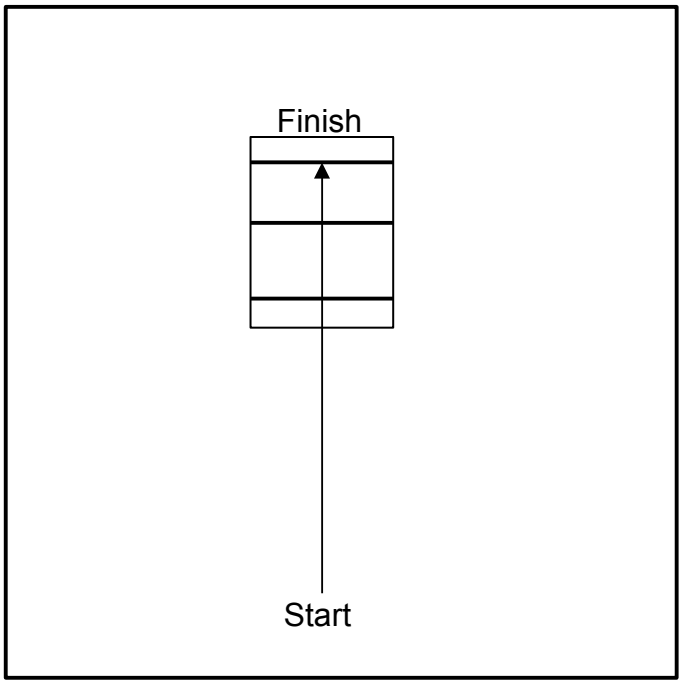
4. Place a move block and change it to stop.



This tells the motors to stop moving quickly. It is like slamming on the breaks.

Stop on the Third Line

Mission:
The robot will use a light sensor to sense when the robot has gone over 5 dark lines and then it will stop.

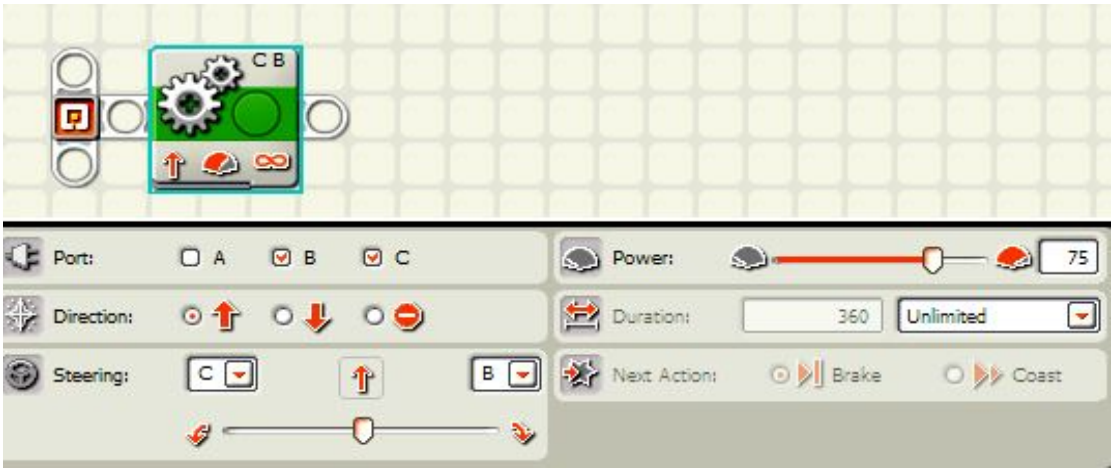


Equipment:
White table top or playing field.
Copy of three lines found at the end of this mission. Place it under the clear plastic sheet.

Sensors:
Light

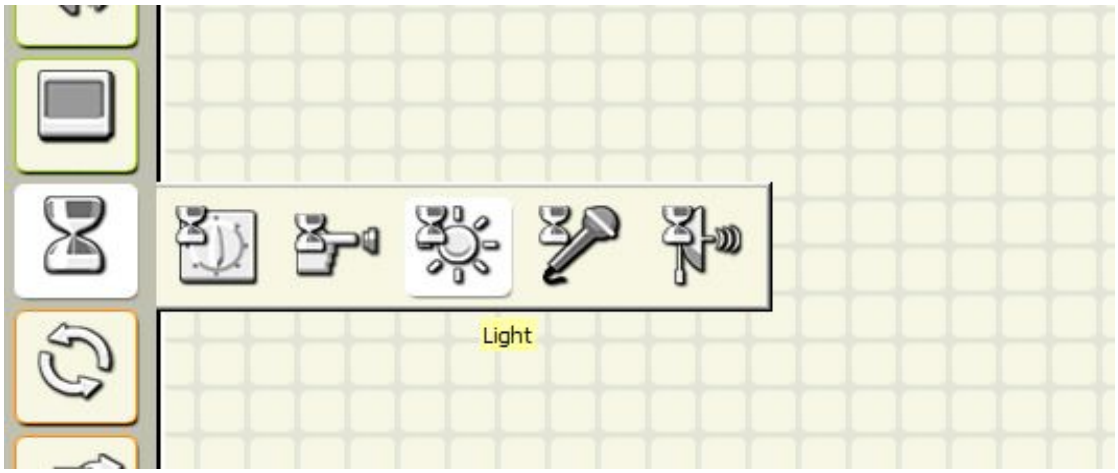
Directions:

1. Place a move block at the beginning of the program line. Set it to unlimited.

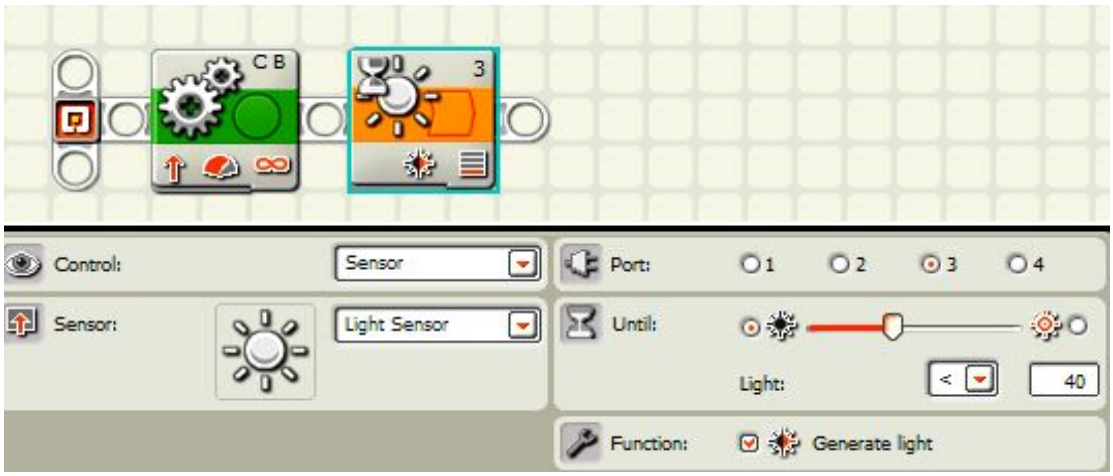


This will make the motors turn continually until a wait block is triggered and the program moves on to the next block.

2. Move the cursor over the wait block and wait until a variety of wait blocks open up. Pick the light wait block. It looks like a shining sun.

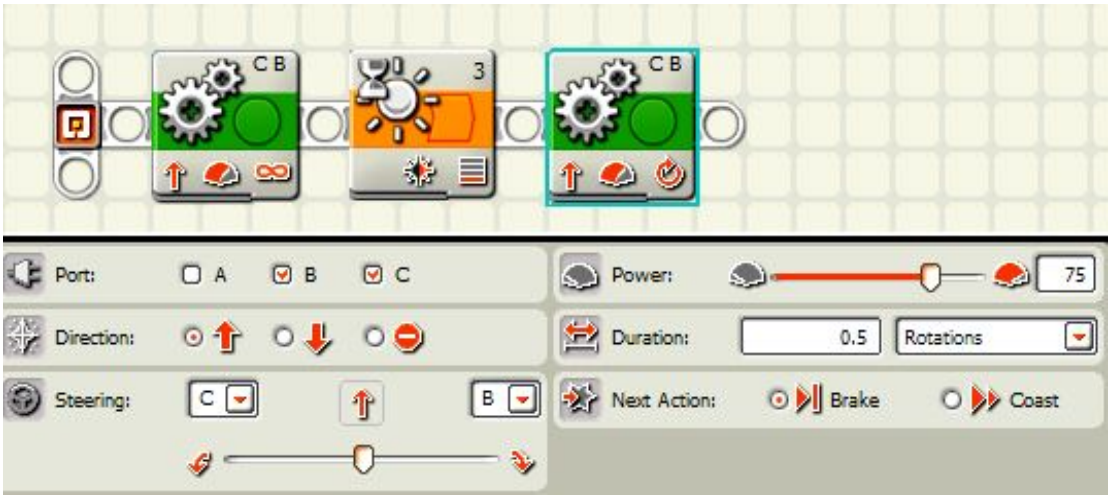


3. Place the light wait block on the bar and set it to less than and set the threshold to 5 points lower than the brightness of the white surface.



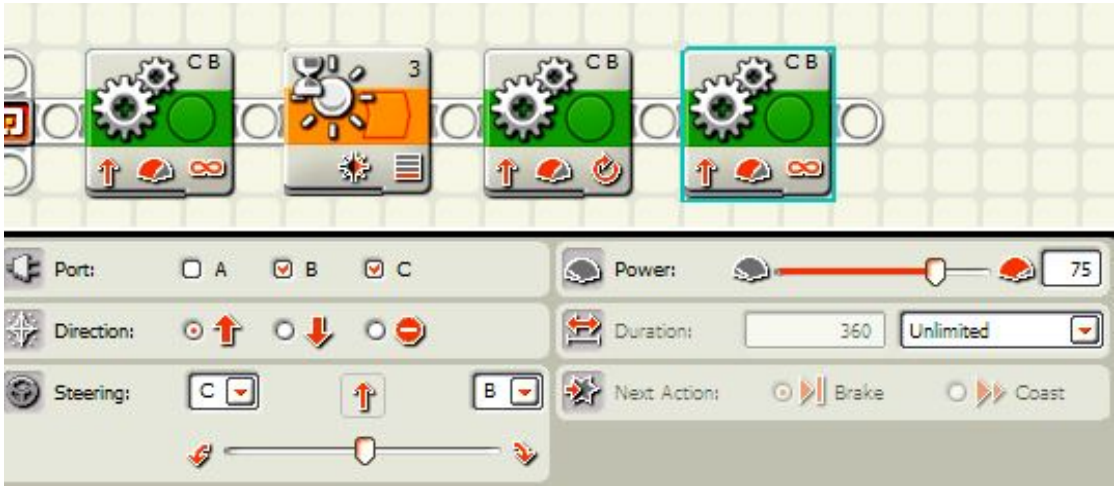
This will make the light wait block be triggered by anything somewhat darker than the white surface.

4. Place a move block on the bar and set it for .5 rotations.



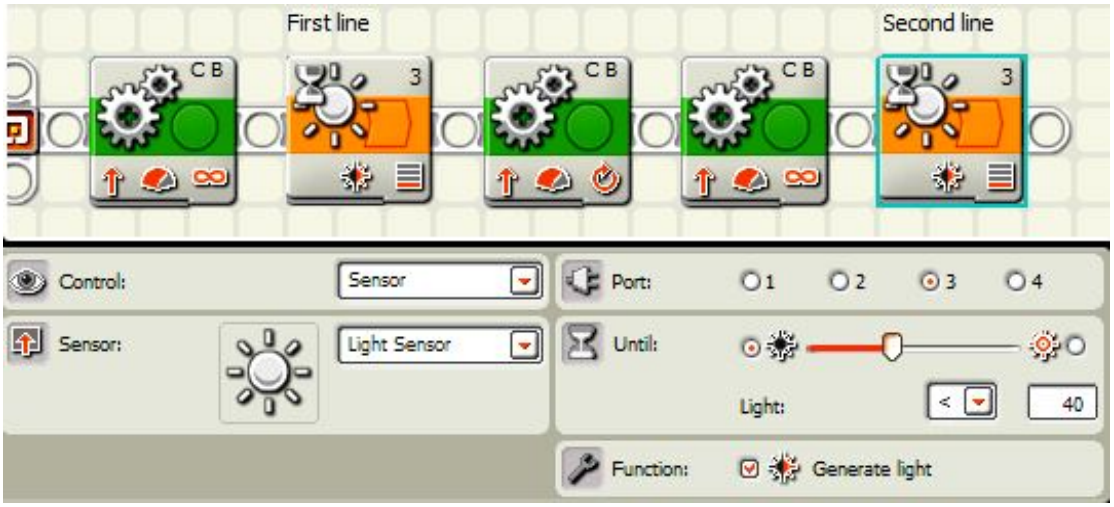
This will make the robot move past the dark tape and have the sensor over the white surface again.

5. Place a move block on the bar and set it to unlimited.



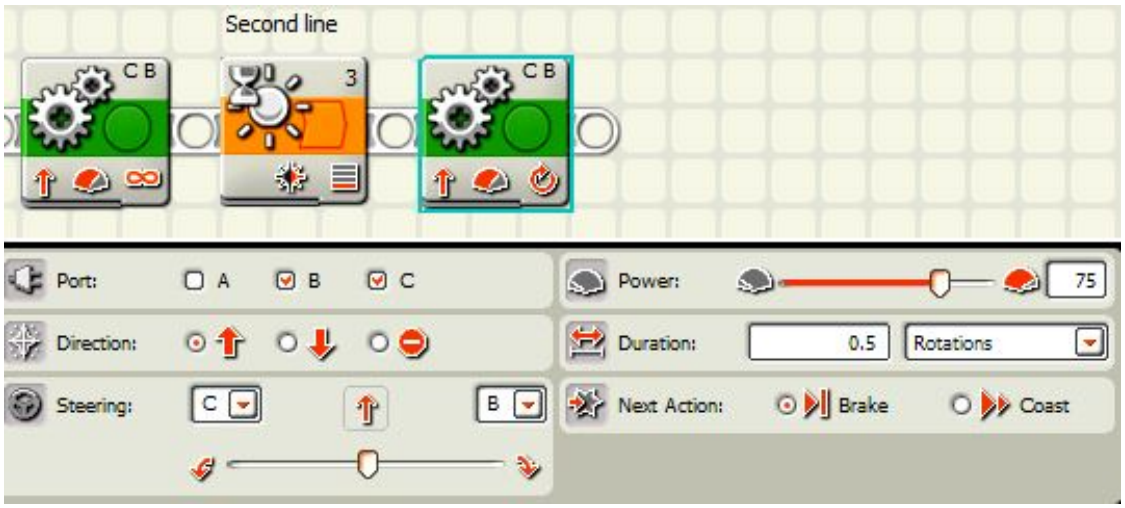
This keeps the robot moving until the light sensor senses the next piece of dark tape.

6. Place another light wait block on the bar and set it to the same as the one in step 3.



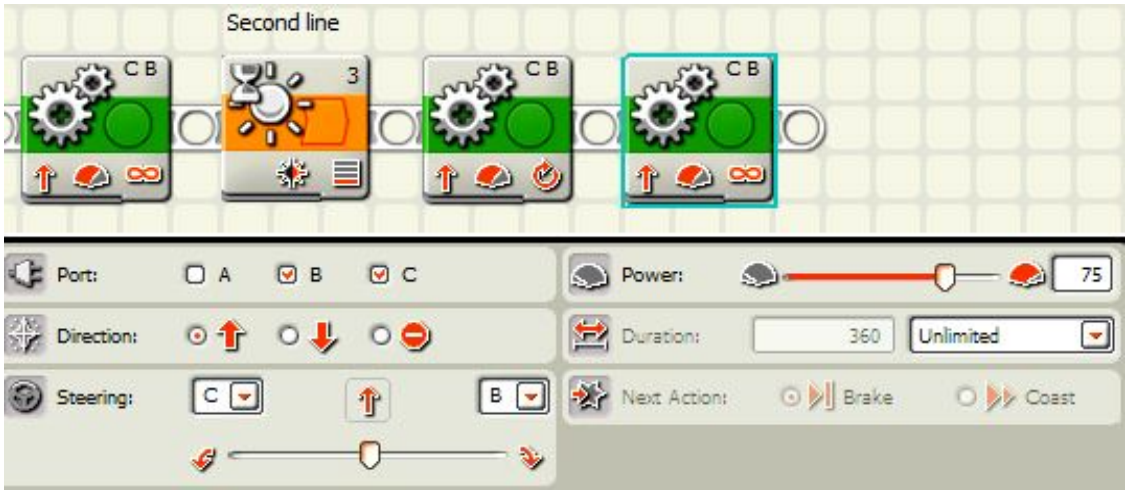
This causes the light block to stop the move block and move on to the move block in the next step.

7. Place a move block on the bar and set it for 0.3 rotations.



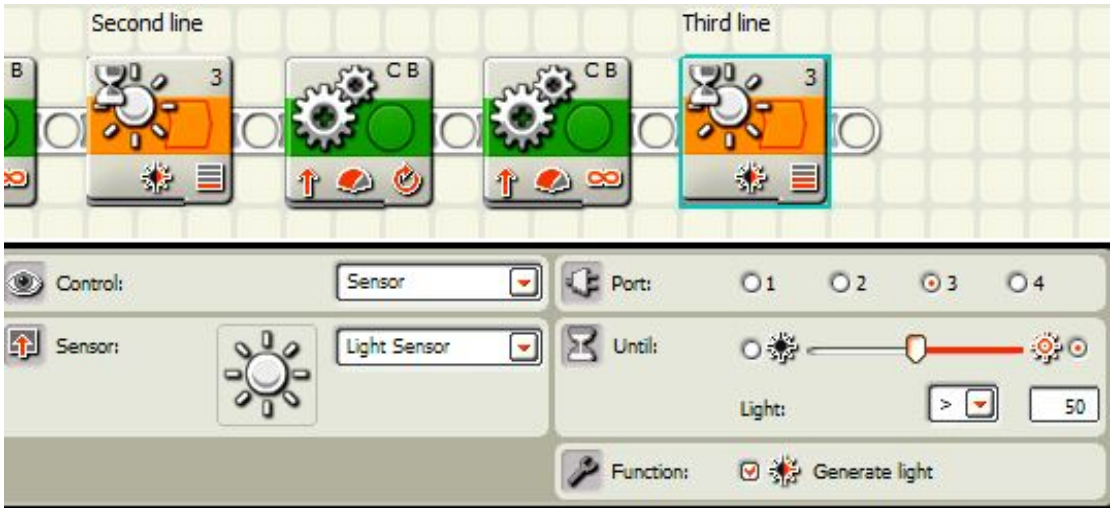
This gets the robot past the second line of tape and gets the sensor over a white surface so it is ready to sense the next line of tape.

8. Place another move block and set it for unlimited.



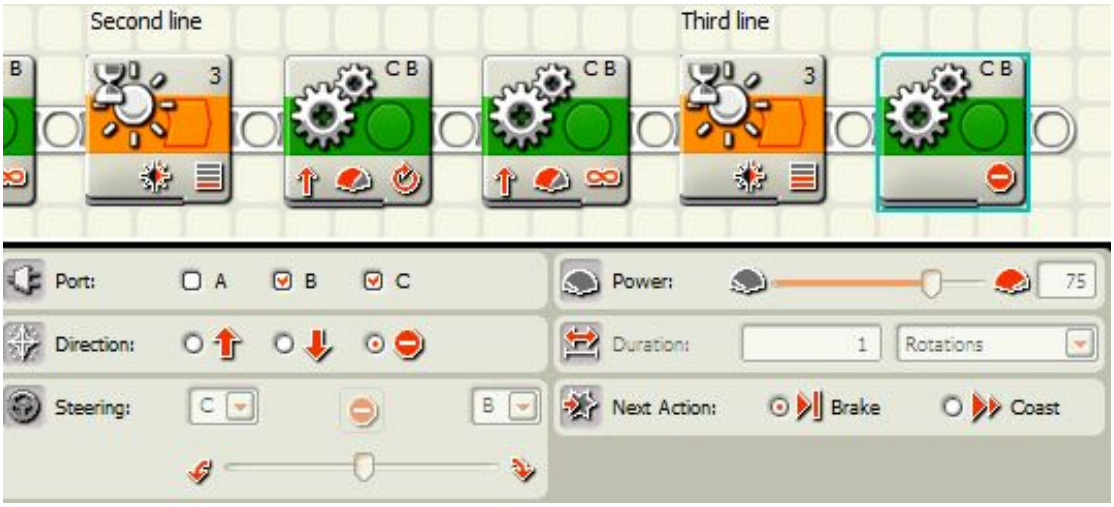
This gets the robot moving to find the 3rd line of tape.

9. Place a light wait block and set it the same as step 3 and 6.



*This will set up the light sensor to find the 3*rd *line of tape and then move the program along to the next step.*

10. Place a move block on the bar and set it for stop.



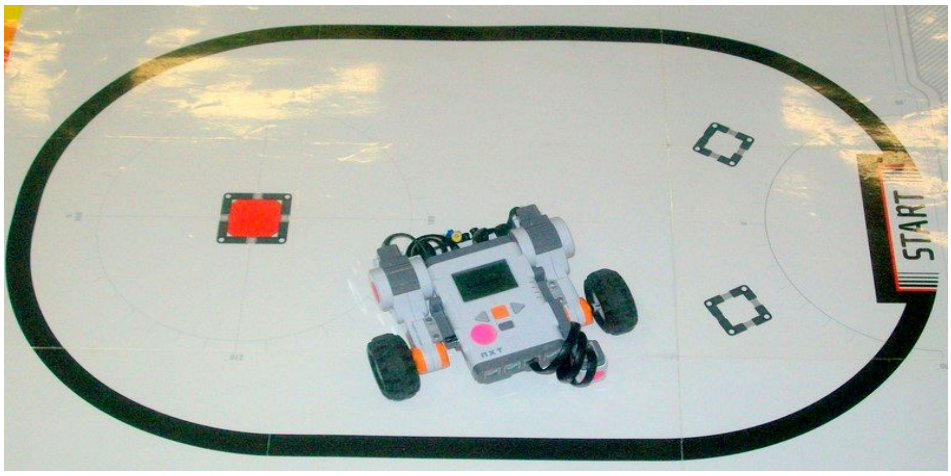
This stops the motors from turning so the robot will stop on the line.

NOTE: Turn to the next page for a set of three lines to use for this exercise. Copy this off and use it under the clear plastic on the practice field. Place it a little past the middle of the field and pick the starting position at random to show that the program is using the light sensor to know when to stop instead of just going a certain number of rotations.

Copy this to use on the mission Stop on the Third Line.

Stay in the Circle

Mission:
The robot will stay in a circle by using a light sensor to sense when it has reached the dark line
of the loop on the practice pad. It will then back up and turn and go forward again.

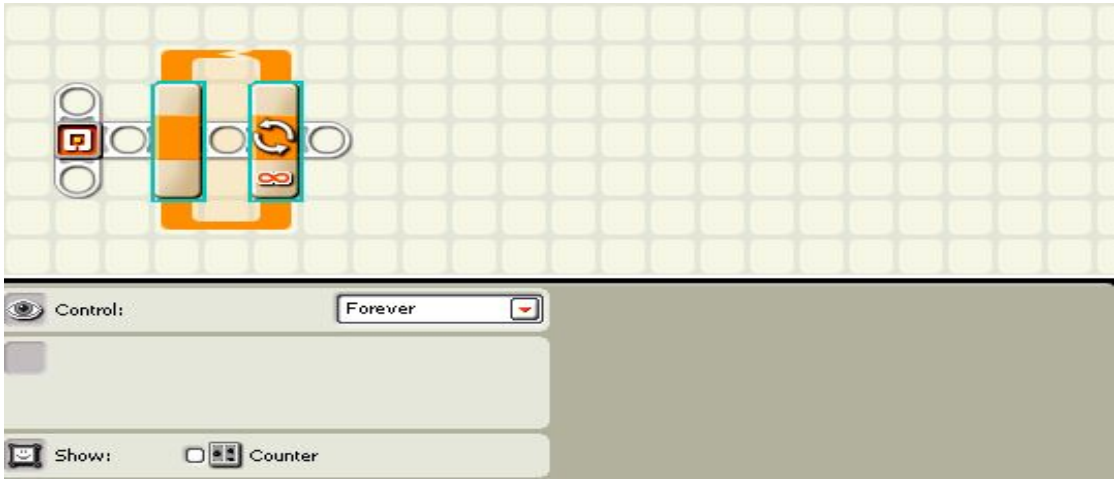


Equipment:
White table top or playing field.
Use the oval on the mat that comes in the Mindstorms kit or make your own oval with blue tape on your own mat.

Sensors:
Light

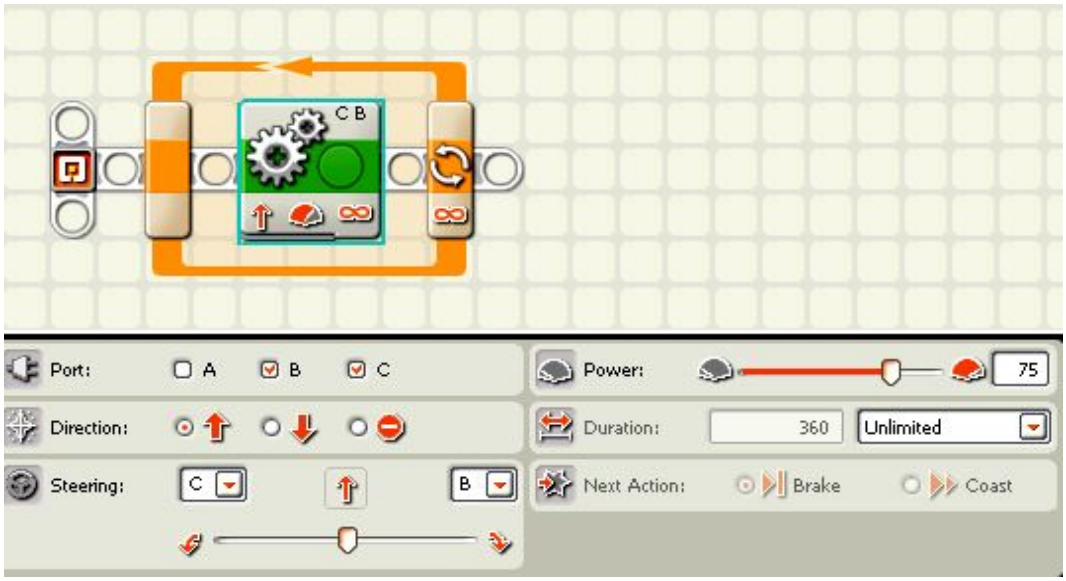
Directions:

1. Place a loop on the line and leave it set at forever.



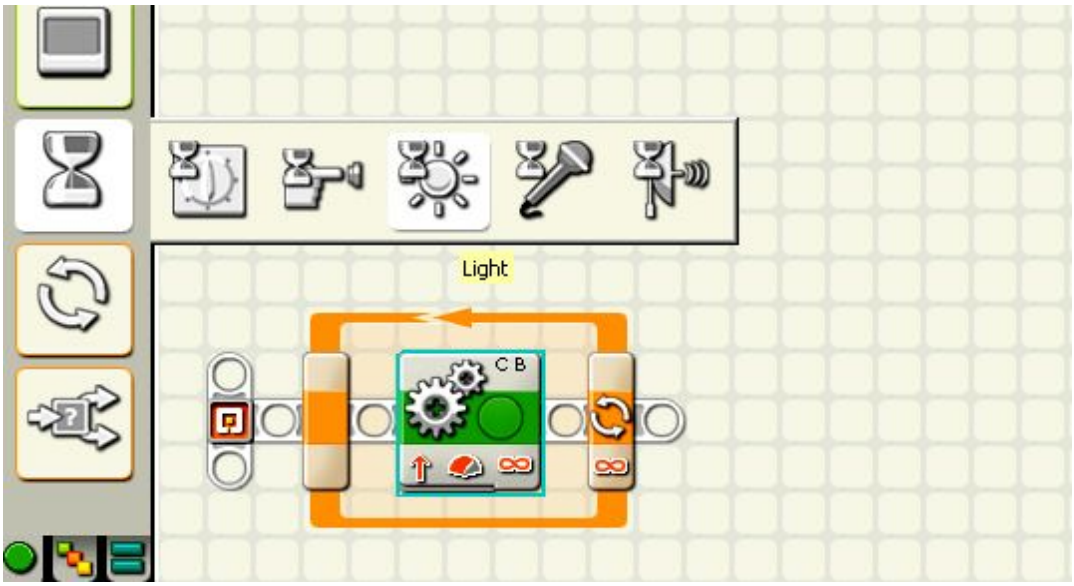
This will make the program loop the code you will put in the loop repeat over and over until you push the dark gray button on the brick to make the program end.

2. Place a move block into the middle of the loop and set it to unlimited.



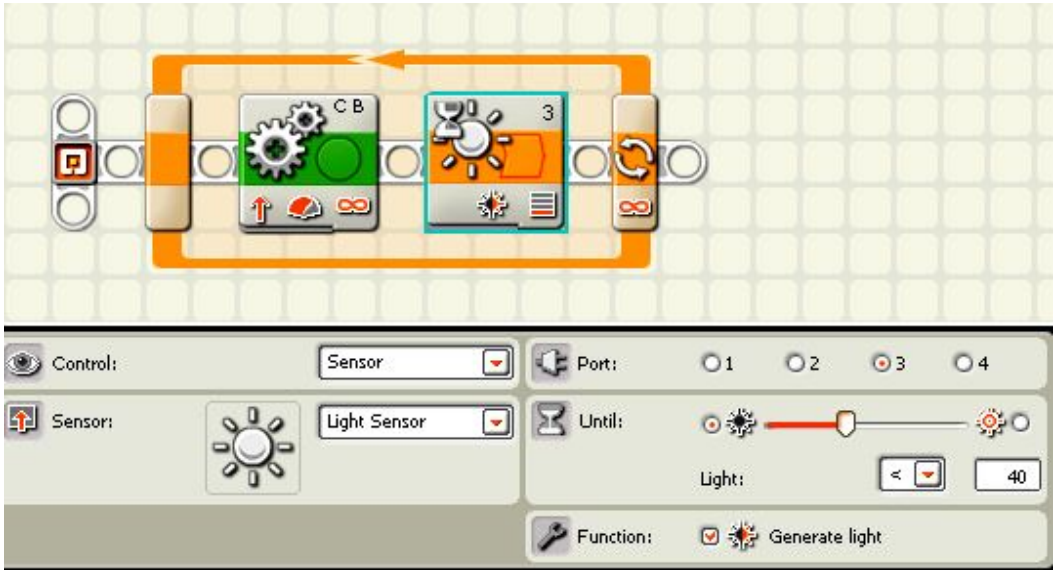
This will make the motors keep turning until a wait block tells it to move to the next block.

3. Move your cursor over the wait block until it slides out to give you a choice of various types of wait blocks. Choose the light wait block.



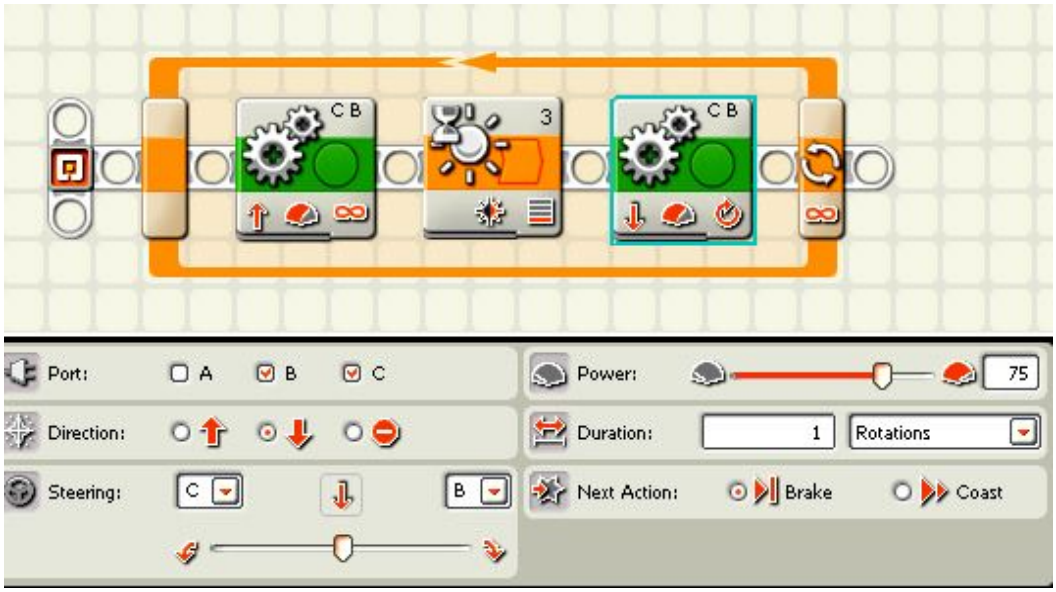
The light wait block is triggered by a reading from the light sensor.

4. Place the light wait block on the bar. Change the greater than sign to a less than sign. Set the threshold number to about five points less than the white setting you took earlier.



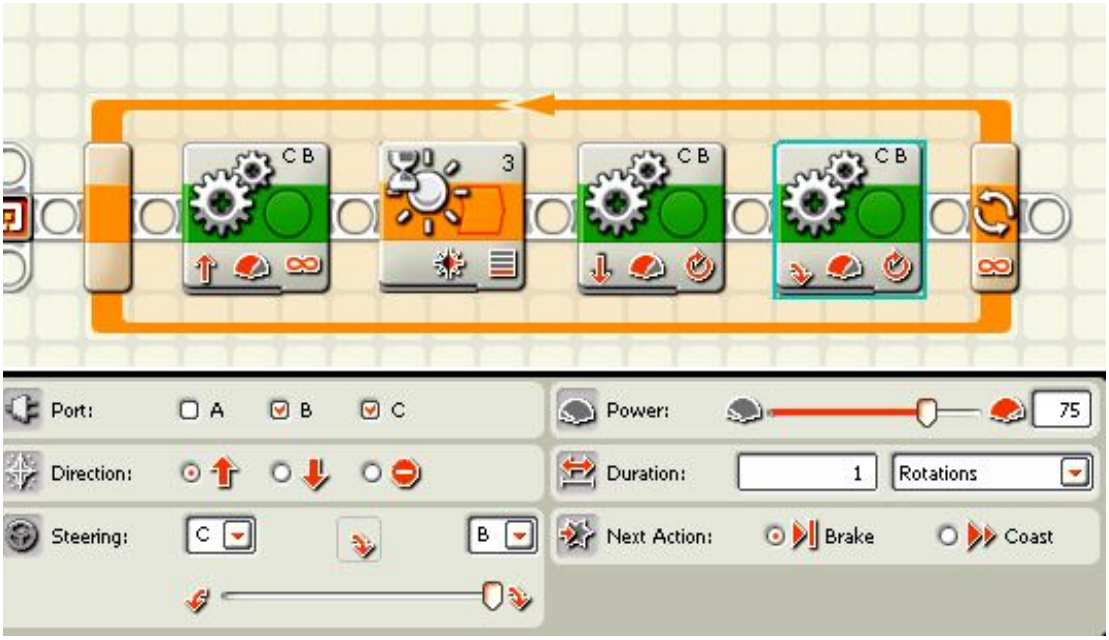
This will cause the block to be triggered by something dark like the oval line.

5. Place a move block after the wait block and set it for reverse. Leave it set for one rotation.



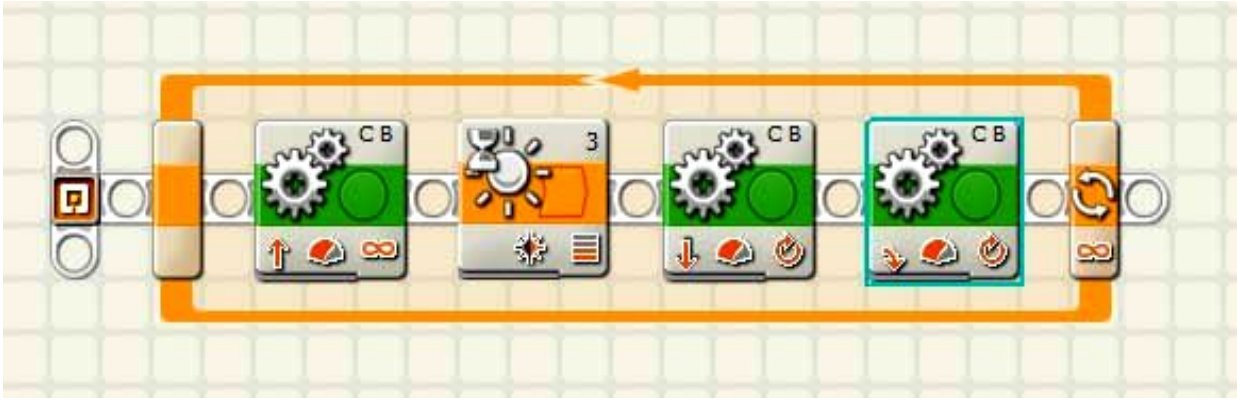
This makes the robot back up away from the line before it turns so that it won't accidentally move the light sensor to the outside of the loop and making the robot drive out of the loop.

6. Set another move block after the last move block, slide the steering slider all the way to one side or the other. Set the number of rotations so that the robot will turn the amount you want it to turn.



This turns the robot so that it will now face a different direction.

7. Here is the whole program.



Notice the loop all around the program. This means that after the robot moves forward, senses the black line, backs up, and turns, it will repeat the whole process over and over.

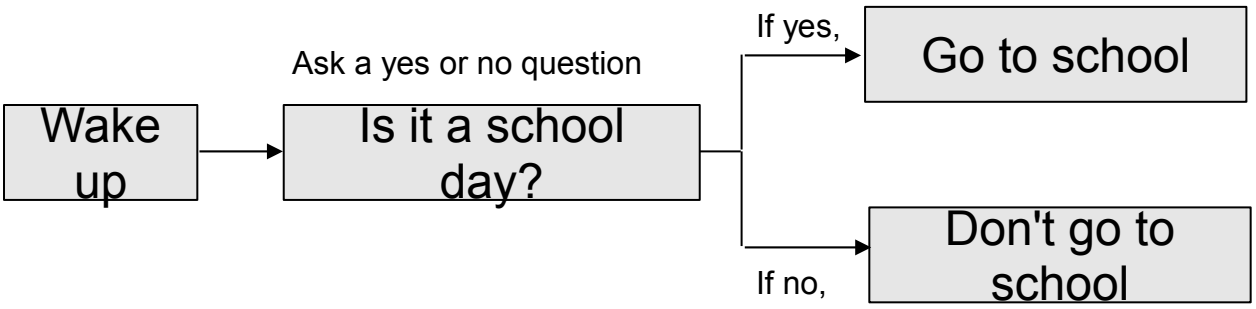
Note:

This can also be used as a basic sumo program where two robots fight to push the other one out of the circle. This can be done on the practice mat or it can be done on a large plywood or particle board circle found at many home improvement centers. It should be about 4 to 5 feet in diameter. Paint it white and paint a black line around the outside edge about three or 4 inches wide to trigger the light sensor. You can also use black duct tape sold at home improvement centers or at large stores like Walmart.

Switch Blocks,

An Explanation

Switch blocks work sort of like wait blocks. A wait block lets the program do one thing under certain conditions and then start a different action if the condition changes. The difference is that a switch block allows the robot to decide between two or more actions depending for some conditions you set. We are only going to show how to use the switch to have only two choices right now. We will set some conditions that have to be matched to make a yes decision and if the condition is not matched, then it is a “no” decision. For instance, below is an simple example of this:



You wake up and you ask yourself if it is a school day. If the answer is yes, you move to the top choice and go to school. If the answer is no, you follow the bottom bar and you don't go to school. That works one time.

Now you could put this switch in a loop so every morning when you wake up you go through the same process over and over again: wake up, ask “Is it a school day?”, if the answer is yes you get up and if the answer is no you don't go to school, then you repeat process over again the next morning.

For example, you could use a touch sensor and that the condition is that the touch senor is pushed because the robot hit a wall.

The switch makes the program make a decision. The conditions for the decision can be all sorts of things. It could be if the light sensor senses something dark or something light. It could be that the touch sensor bumps into something, It could be that the ultrasonic sensor senses something close or something far away. It could be the sound sensor senses a loud or soft sound or stops hearing a loud or soft sound.

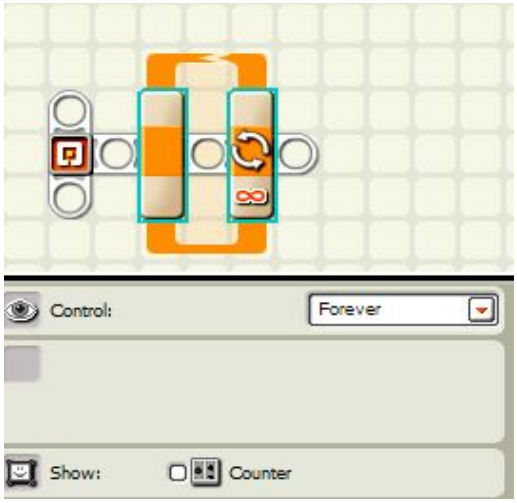
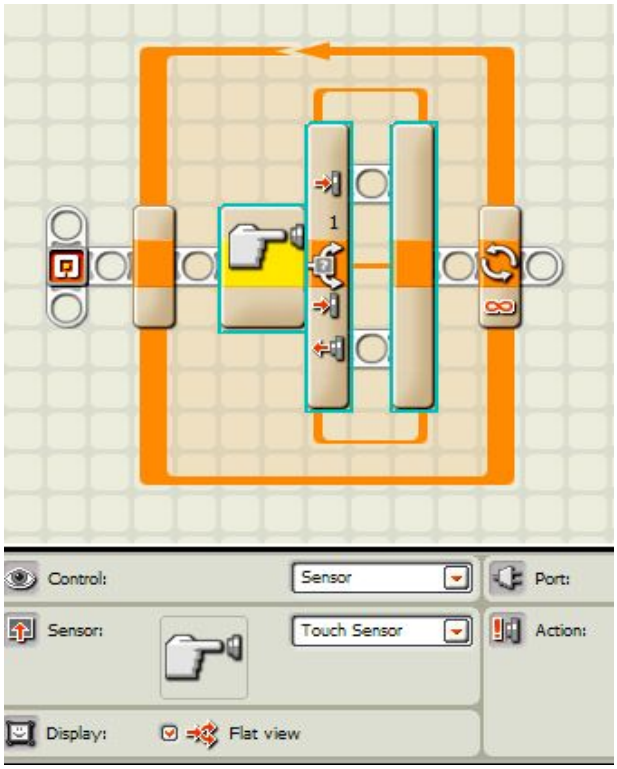
Follow the Line

Mission:
The robot will use a light sensor facing down to follow the dark line on the practice pad indefinitely until stopped.

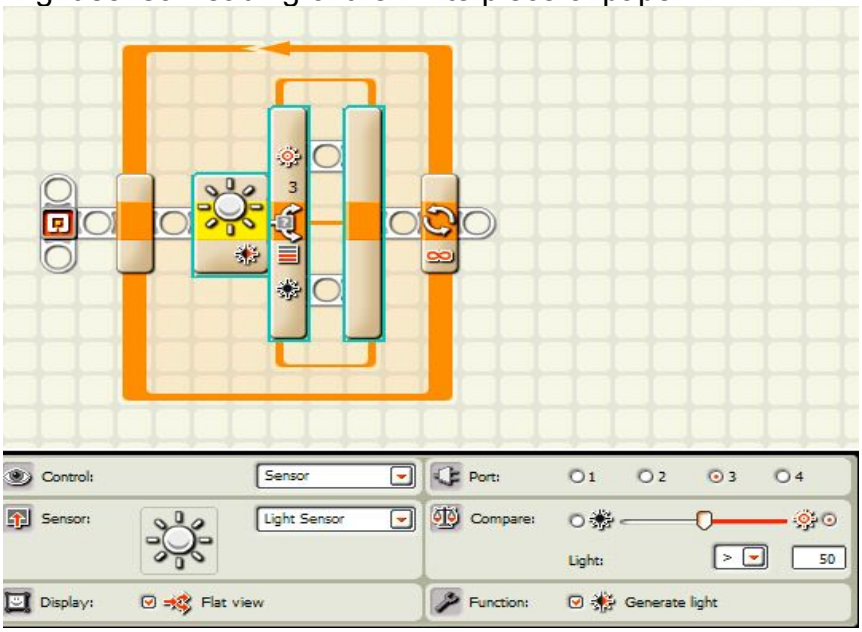
Equipment:
White table top or playing field.
Practice pad oval that came with the kit or blue or green painter's tape to make an oval.

Sensors:
Light

Directions:
Attach the light sensor so it is at least two pennies above the surface.

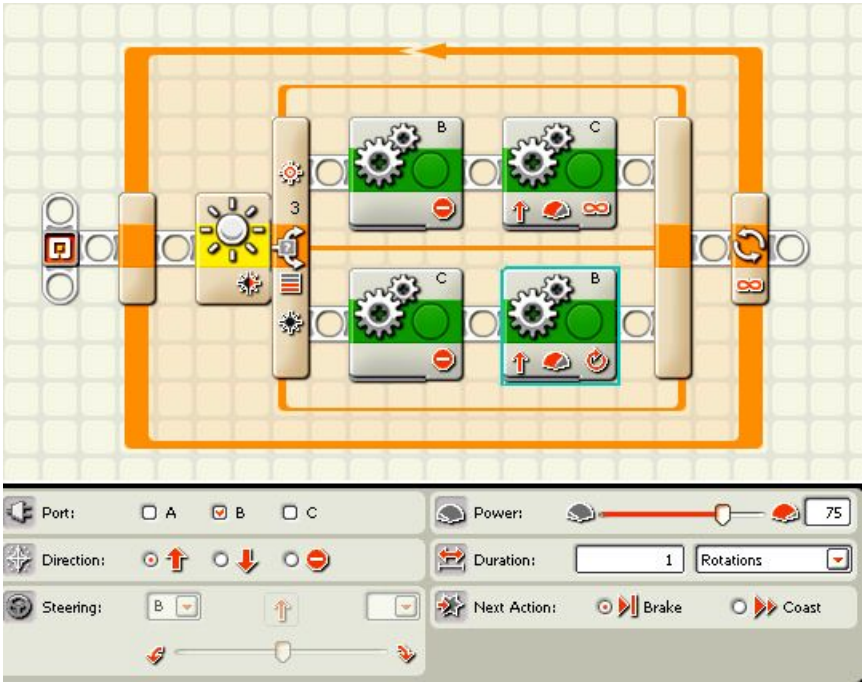
1. Get a Loop block and place it at the start.	2. Get a switch block and put it in the loop.
	
The loop makes the program inside the loop repeat over and over.	

3. Change the Touch Switch to a Light Switch. Set the sensitivity to be about 5 points less than your light sensor reading of the white piece of paper.

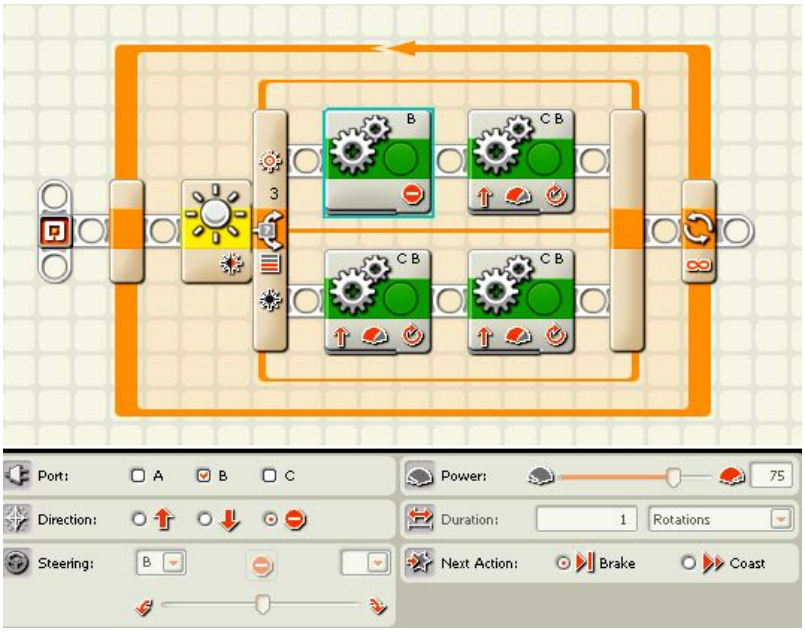


Making the light setting 4 or 5 point lower than the white reading makes the switch change when the light sensor is only part way on the black line.

4. Put two Move blocks on each line.

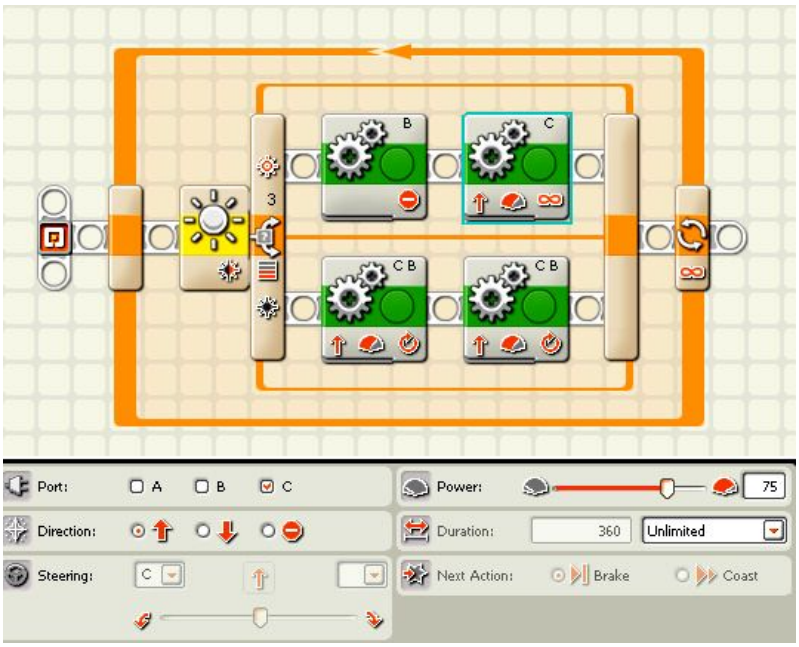


5. On the top line and in the first block set to Port B and to stop.



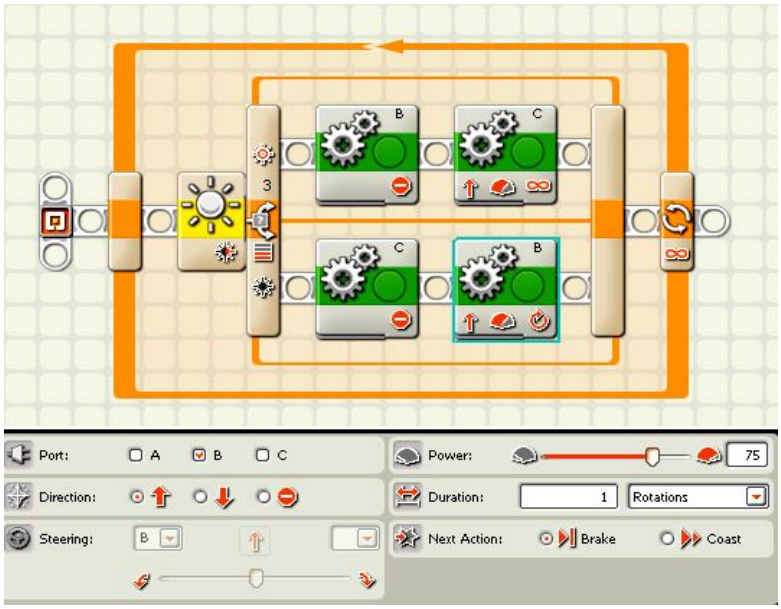
This makes wheel B stop.

6. Still on the top line, set the second block to Port C and make it unlimited.



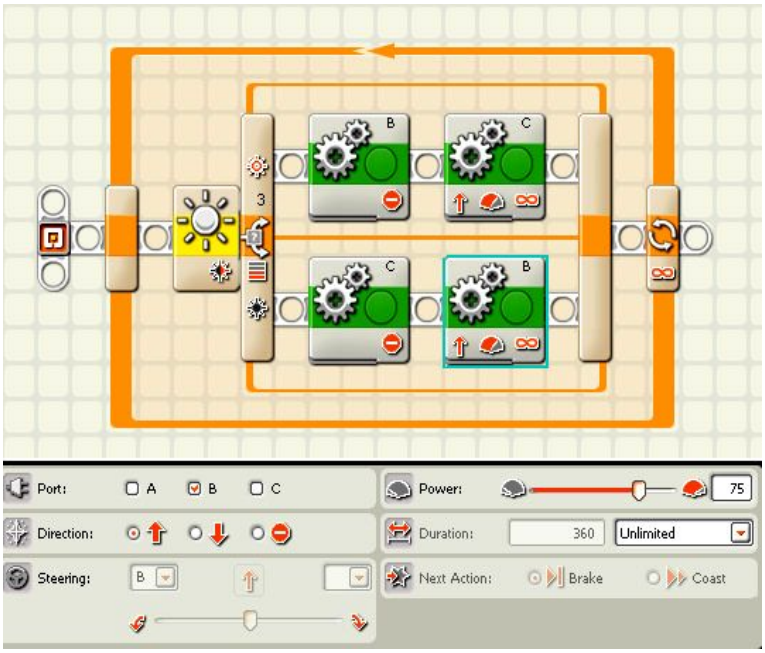
This makes wheel C keep moving until the switch changes.

7. On the bottom line, set the first block to Port C and to stop.



This makes wheel C stop.

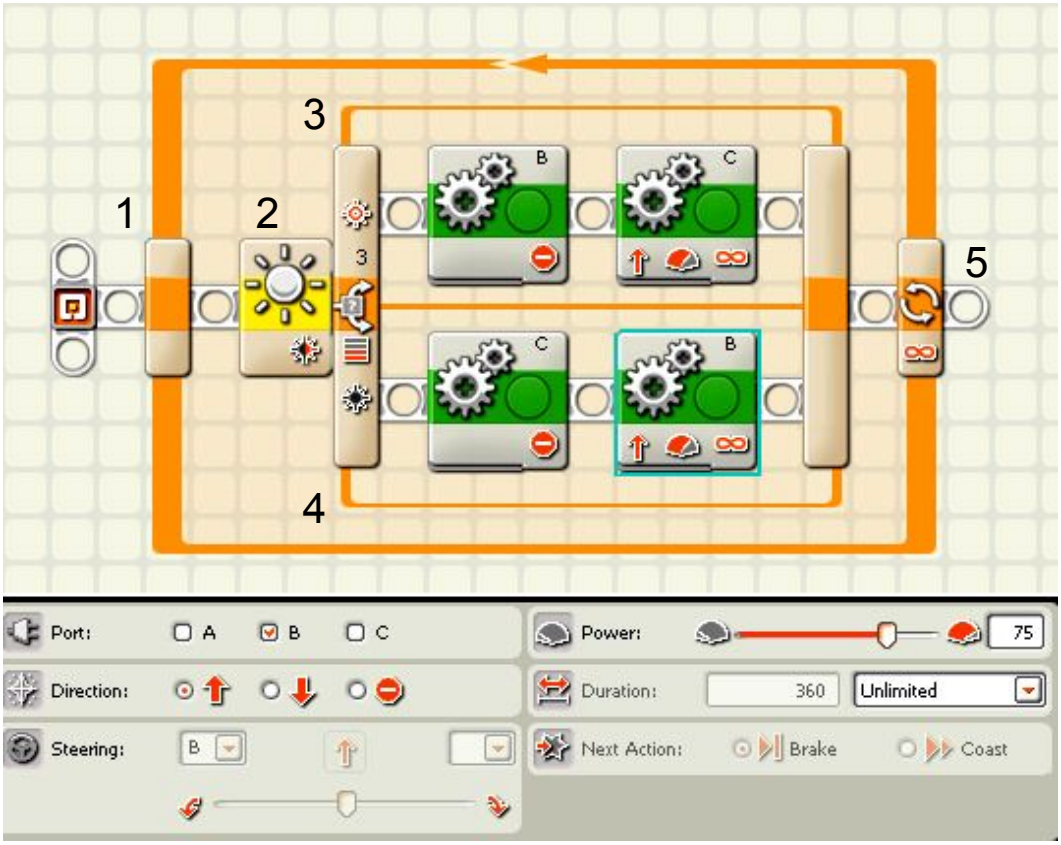
8. Set the second block on the bottom line to Port B and to unlimited.



This makes wheel B keep going until the switch changes to a different line.

To review,

1. The loop makes everything happen over and over.
2. The switch decides if the brightness of the light is bright enough to say yes, it is brighter than the level you set or no it is not.
3. If it is brighter, the program goes to the top level and stops the wheel connected to port B and makes the wheel connected to motor C keep going until something changes.
4. If the answer is no, it is not brighter than the level you set, the program takes the lower line of the switch and stops the wheel connected to port C and makes the wheel connected to port B keep going until something changes.
5. The loop repeats everything forever until the program is stopped.



Secret of success: You may need to adjust the sensitivity of the switch block higher or lower to make it respond to the dark line but not respond to slight changes in the lighting. Also, keep the robot moving at ½ to ¾ speed so the sensor will not move to the other side of the black line before the robot has a chance to turn away from the line.

Twice Around the Loop

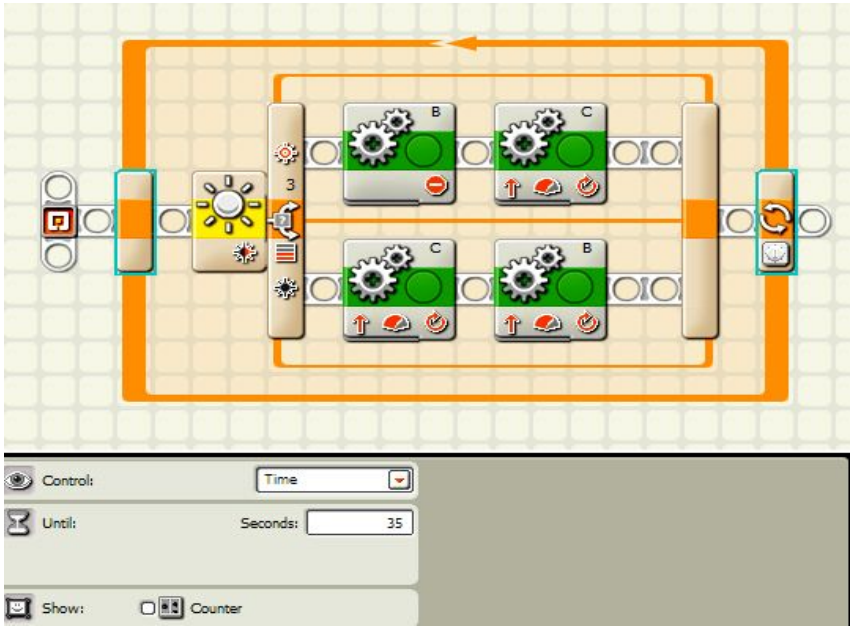
Mission:
The robot will use a light sensor facing down to follow the dark line on the practice pad for two times around the loop.

Equipment:
White table top or playing field.
Practice pad oval that came with the kit or blue or green painter's tape to make an oval.

Sensors:
Light

Directions:
Attach the light sensor so it is at least two pennies above the surface.

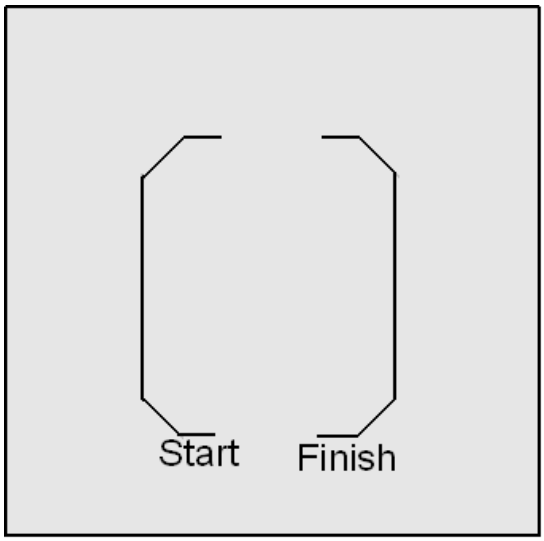
Use the program from Follow the Line and click on the loop. On the lower left, you see Control. Click on the drop down menu and change it to Time. Set it for about 35 seconds.



You will need to adjust the amount of seconds the loop is given so the robot will go around exactly two times. Seconds are not always the best way to time things since well-charged batteries will make the robot move faster than nearly dead ones, but Time works the best in this situation.

2 Halves of a Circle

Mission: The robot will start at the word start and use the line following program from the last assignment to go around the loop on the test pad two times and stop where it started give or take one width of the robot.

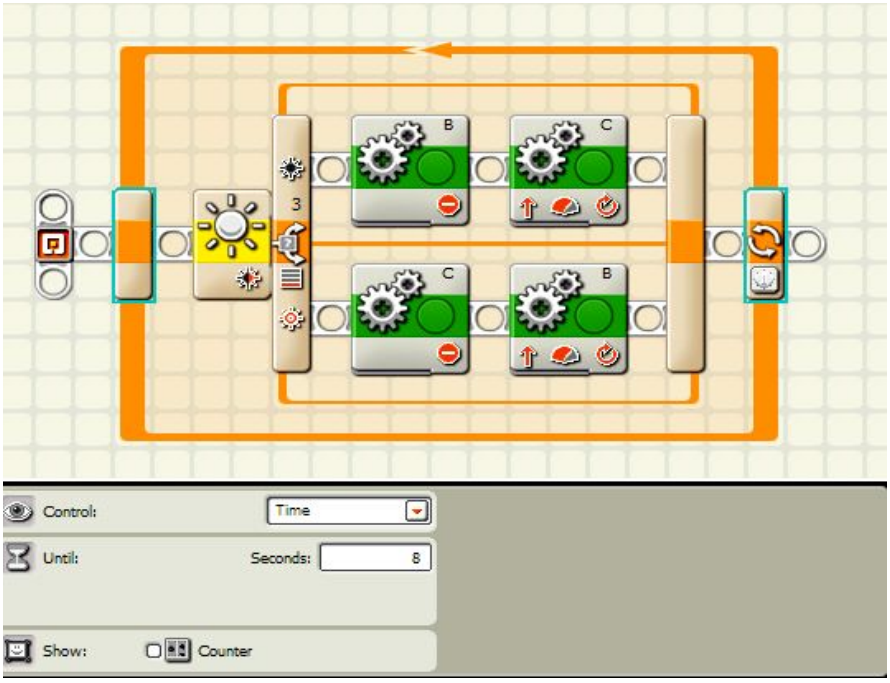


Equipment:
White table top or playing field.
Practice pad oval that came with the kit or blue or green masking tape to make an oval.

Sensors: Light

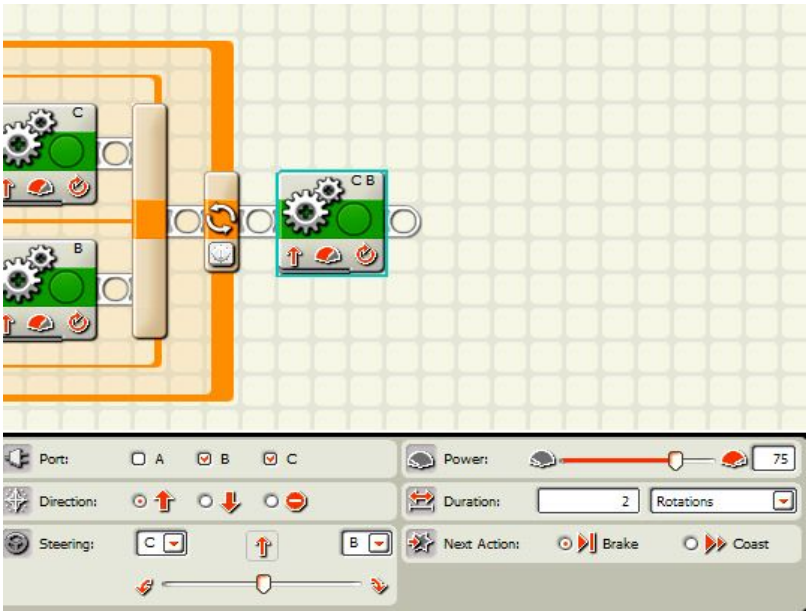
Directions: Attach the light sensor so it is at least two pennies above the surface.

1. Use the light following program explained in the Follow the Line mission earlier. Change the loop to time and set it for about 8 seconds.



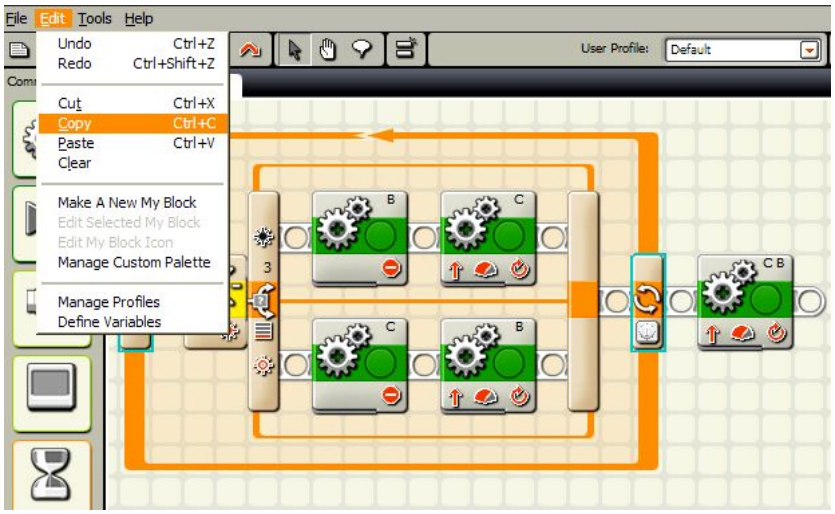
This makes the robot follow a dark line for 8 seconds which should be long enough for the robot to get a little over half way around the oval on the practice pad.

2. Place a move block after the follow the line loop and leave it set for about two rotations.



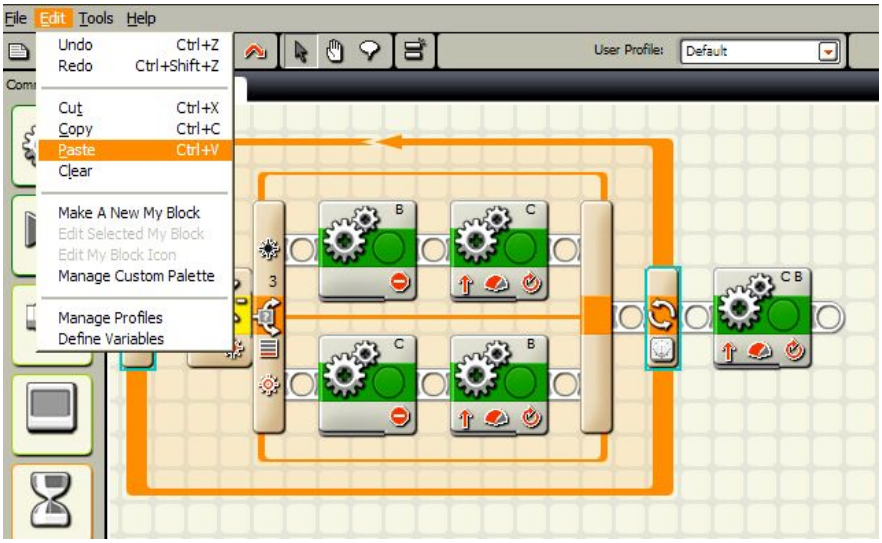
This makes the robot move away from the first oval and over by the other oval. It is best to aim the robot to be a bit inside the loop so that when the next loop starts looking for the line, it will find it and start following it.

3. Click on the loop and go to edit to copy it.

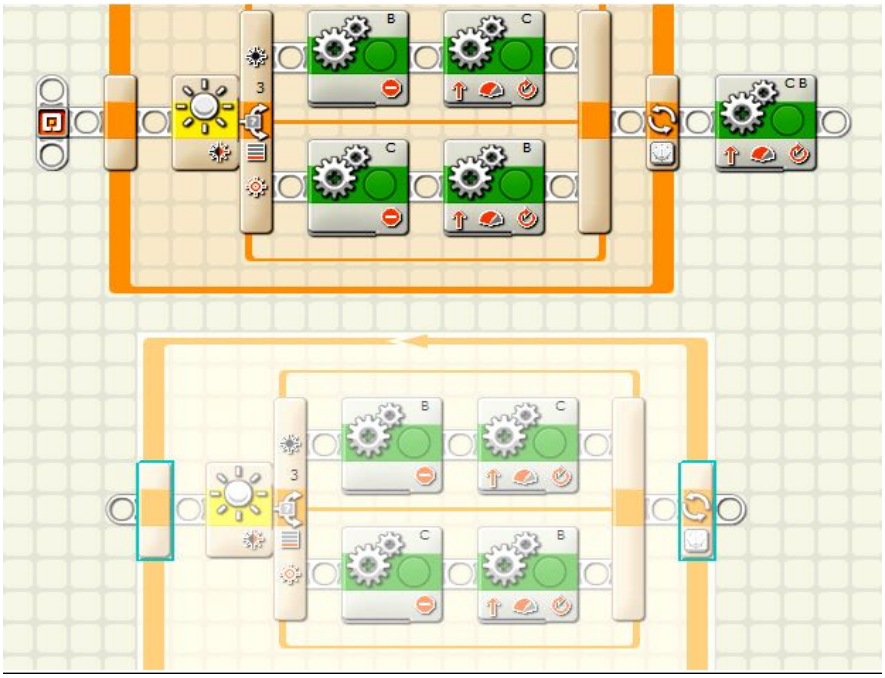


You can redo the loop piece by piece, but it is easier to copy and paste it. It is a good time-saver.

4. Go back to edit and click paste.

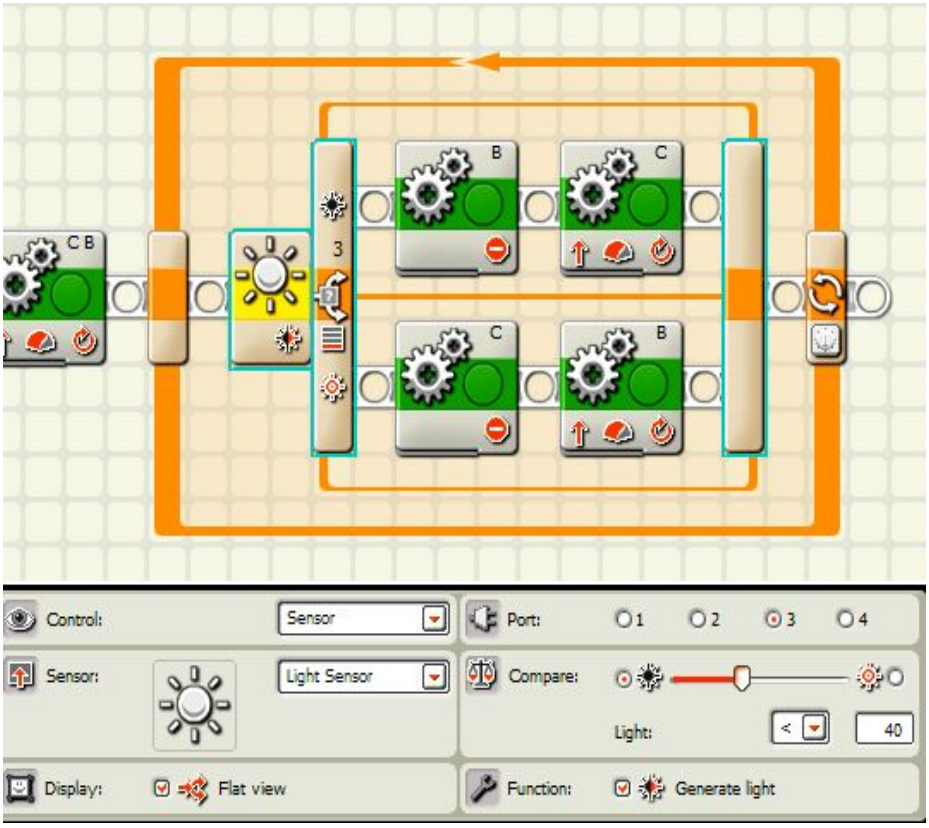


6. The loop is pasted into the program but not on the program line. You will need to click on it and move it to the end of the bar.



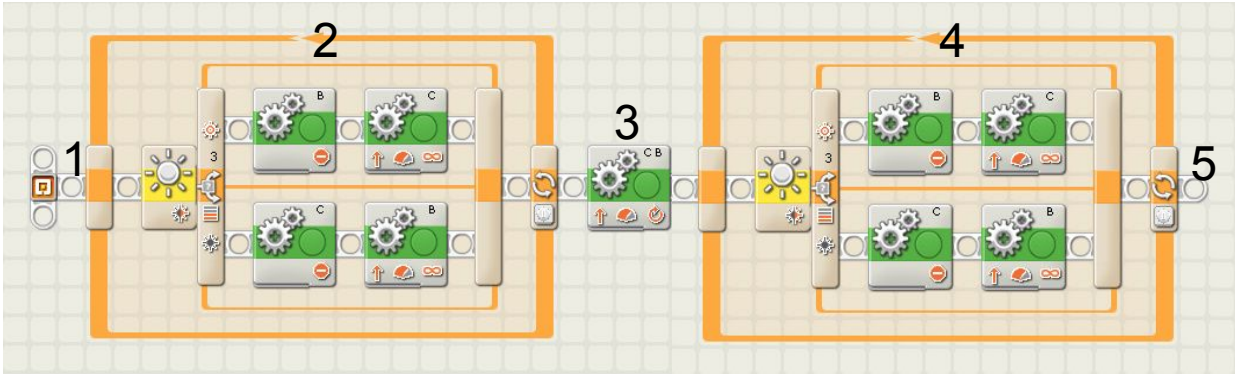
Notice the loop that the program pasted is lighter or "ghosted." This means that it is not part of the program since it is not on the bar. If you were to run the program right now, it the program will not run this section yet.

7. Now you have a program that has a loop, a move bar, and another loop.



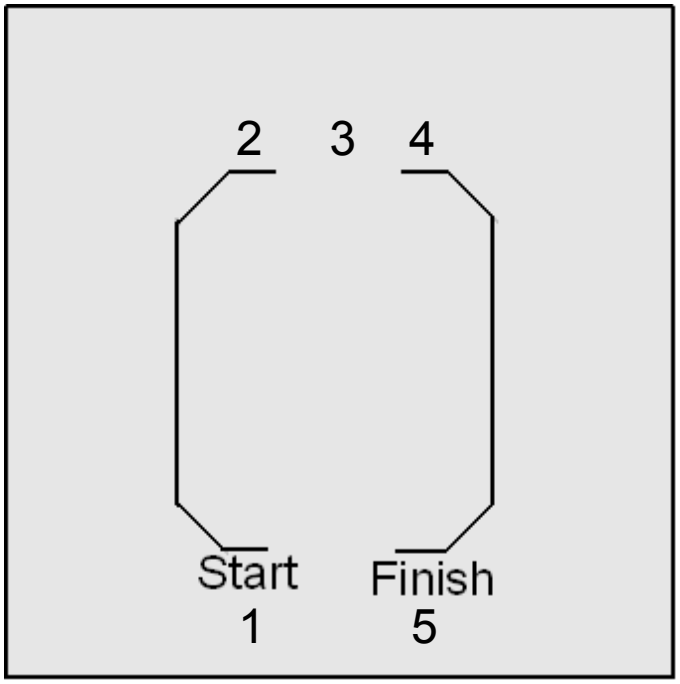
Now the second loop is part of the program so it is now the same brightness as the other parts. The program will now do this loop as part of the program.

8. The whole program looks like this.



The program will run like this:

1. *The robot starts on the word Start on the practice pad*
2. *The robot follows the line for a little over half the oval until it gets a little past the top part.*
3. *It then cuts across the top to a little inside the second oval.*
4. *Then the second loop in the program takes over and the robot follows the line around the line of the second loop.*
5. *It stops near the word "Start" on the practice pad (Finish on the illustration below).*



Smile and Frown

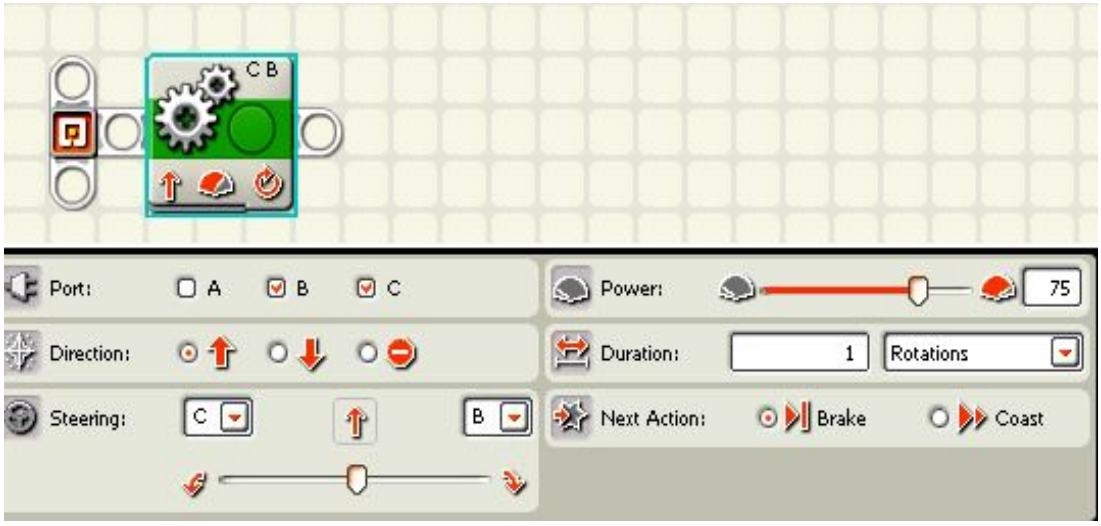
Mission: the robot will go forward one rotation, have a smile appear on the screen, wait 2 seconds, back up one rotation, make a frown appear on the screen, and then stop.

Equipment:
none

Sensors:
none

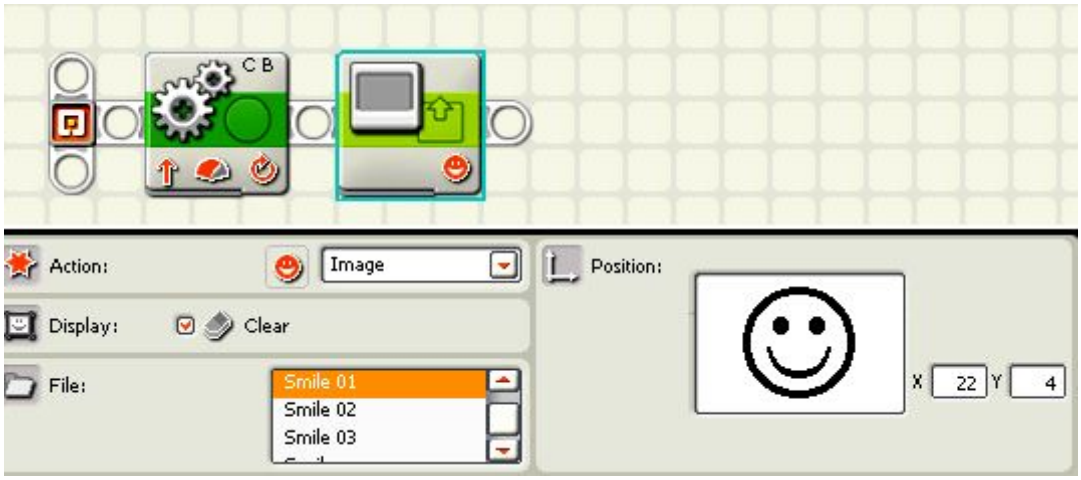
Directions:

1. Place a Move block and leave it at one rotation.



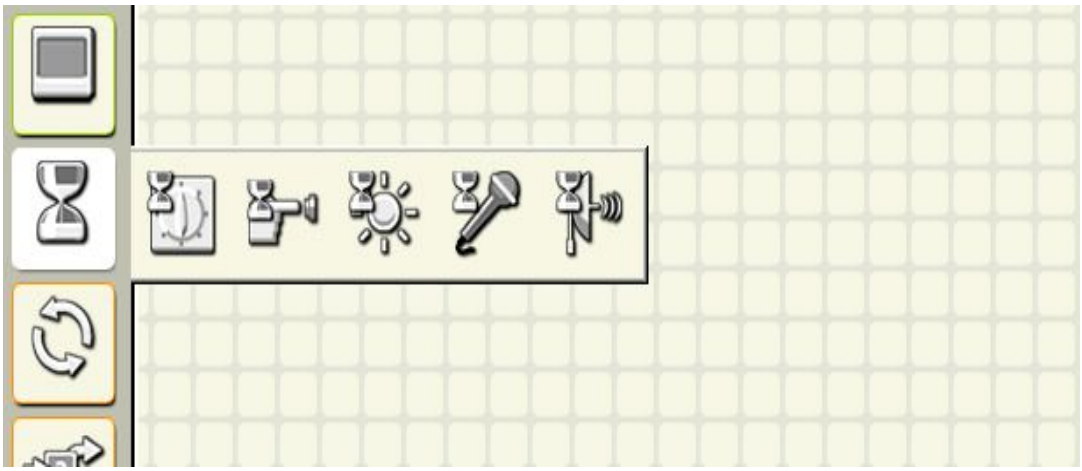
This will make the robot move forward one rotation

2. Place a Display block and leave it at Smile 1.

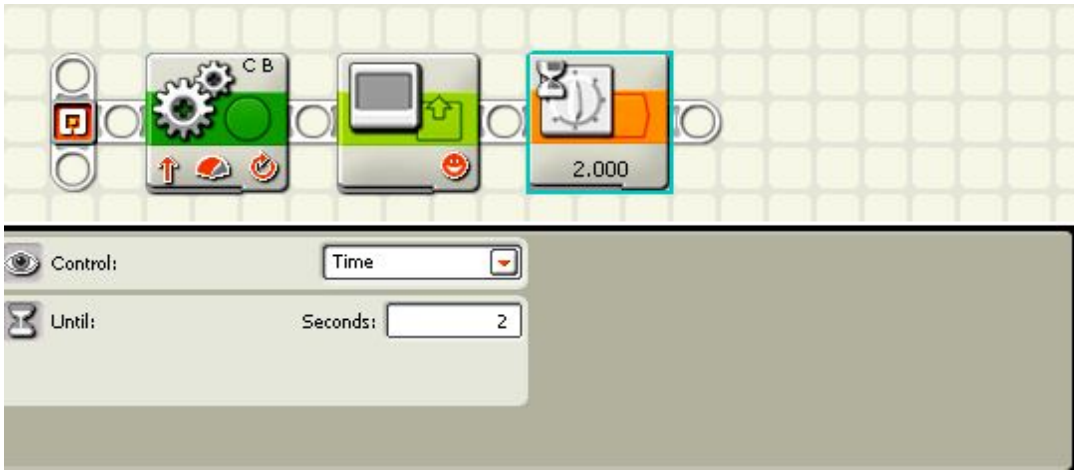


This will make a smiley face appear on the screen when the program runs.

3. Move the cursor over the Wait block (it looks like an hour glass) and several different types of Wait blocks will appear. Click on the first one that looks like a kitchen timer.

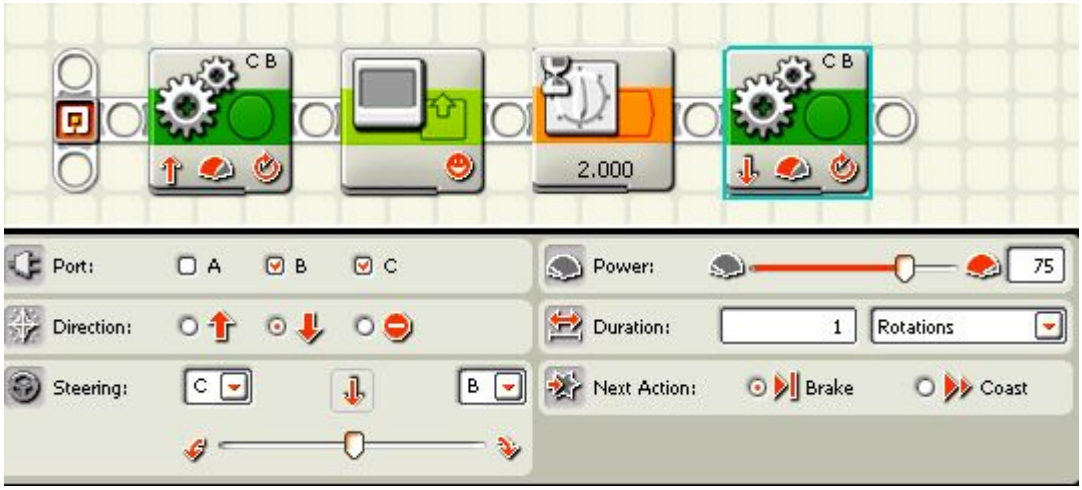


4. Place a Time Wait block and set it to two seconds.

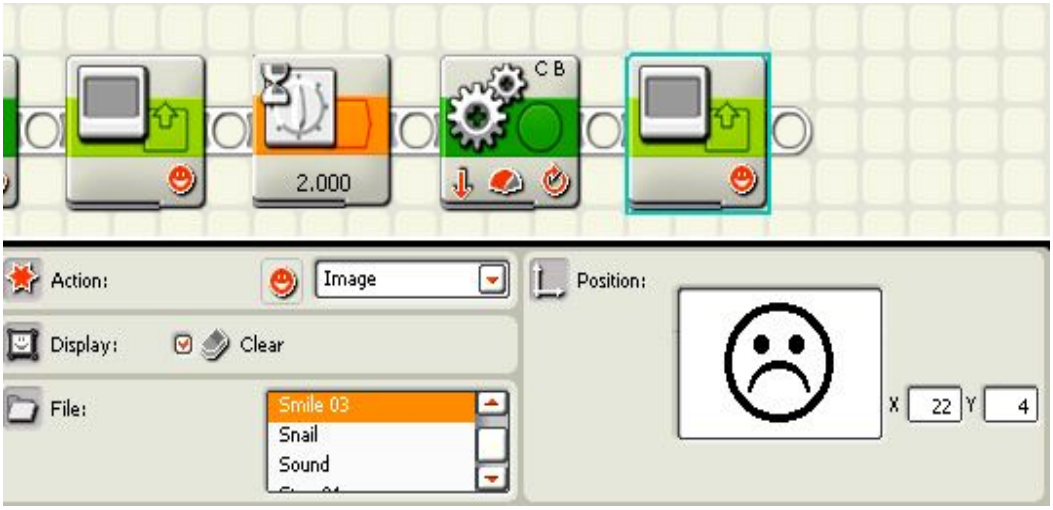


This makes nothing happen for 2 seconds after the picture appears.

5. Place a Move block and set it to reverse.

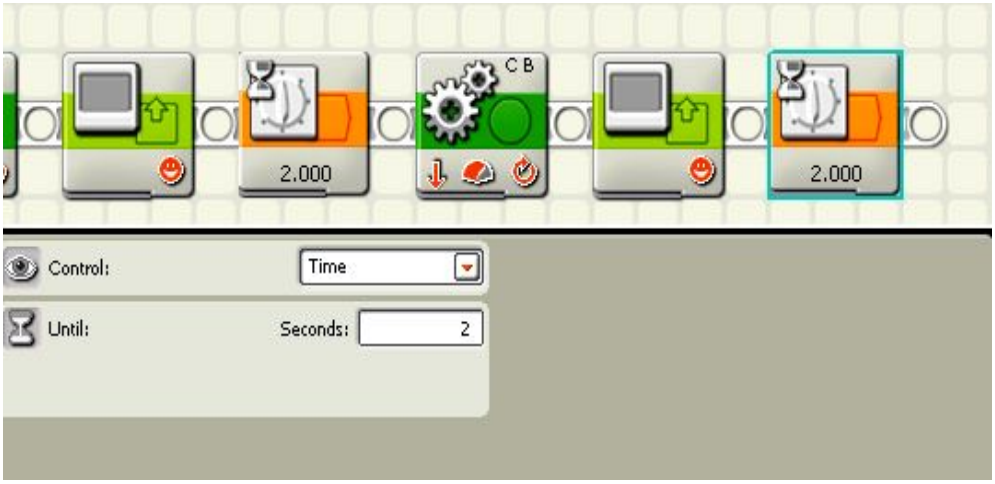


6. Place a Display block on the bar and set it to Smile 3.



This will change the picture to a frowning face.

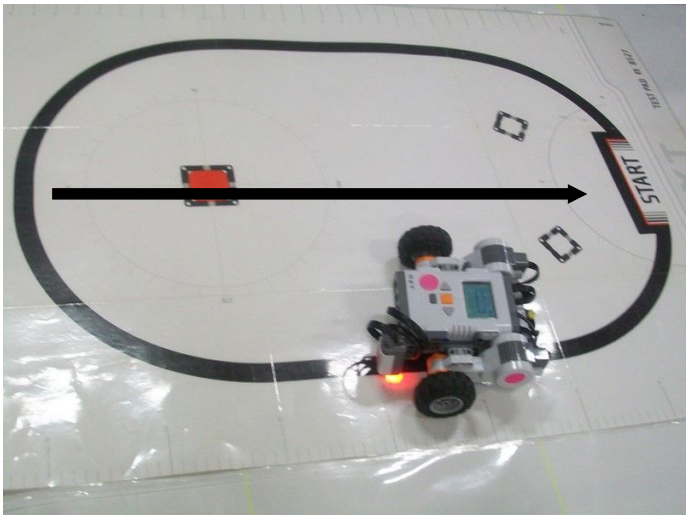
7. Set a Wait block and set it to 2 seconds.



This keeps the program running for 2 seconds after the frowning face appears. If you don't put this here, the program will end a split second after the second picture appears and you won't see it.

Half Circle and Back

Mission: The robot starts at the lower left hand part and will travel a half circle up and around the first line using the light sensor to follow the line. The robot will cut across the top using a move block, and then it will resume using the line following program to come down the other side and

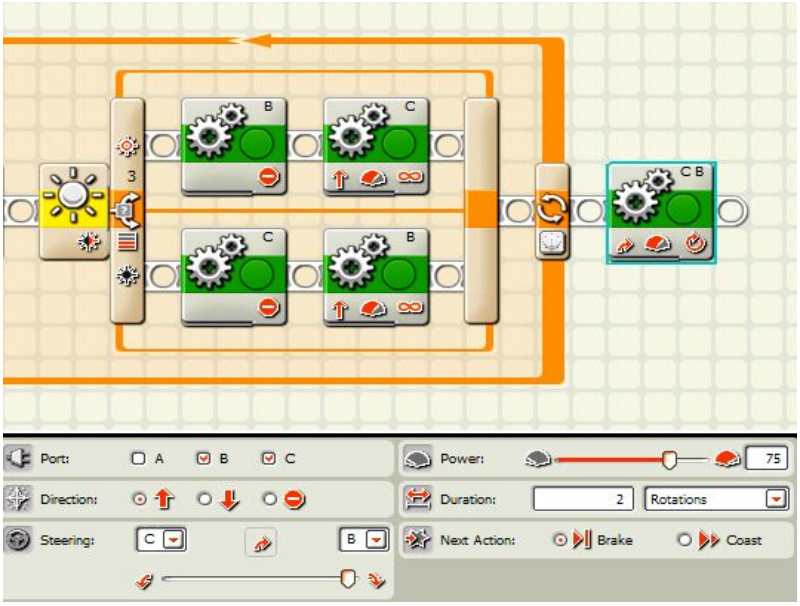


Equipment: Practice pad or blue painter's tape to layout the design.

Sensors: Light

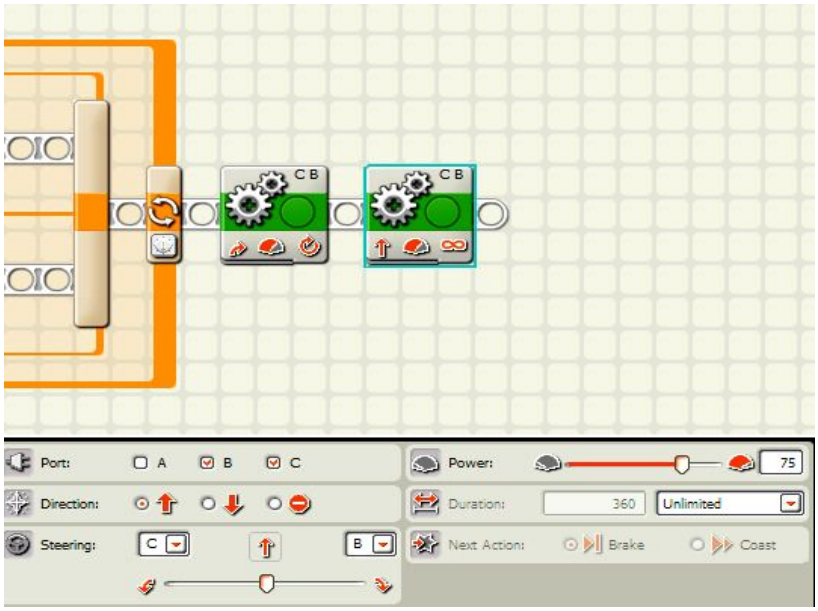
Directions:

1. Start with the Follow the Line program. Then add a move block after that and move the steering slider all the way to the side.



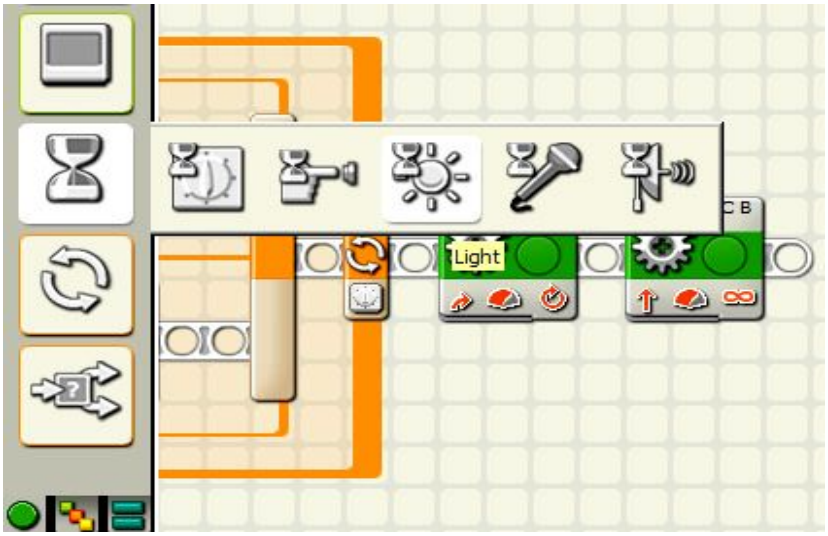
The Follow the Line program makes the robot follow the first half of the loop. You will need to set the loop for about 7 seconds. After the line following program ends, the program moves to the move block which turns the robot.

2. Place move block on the bar and set it to unlimited.

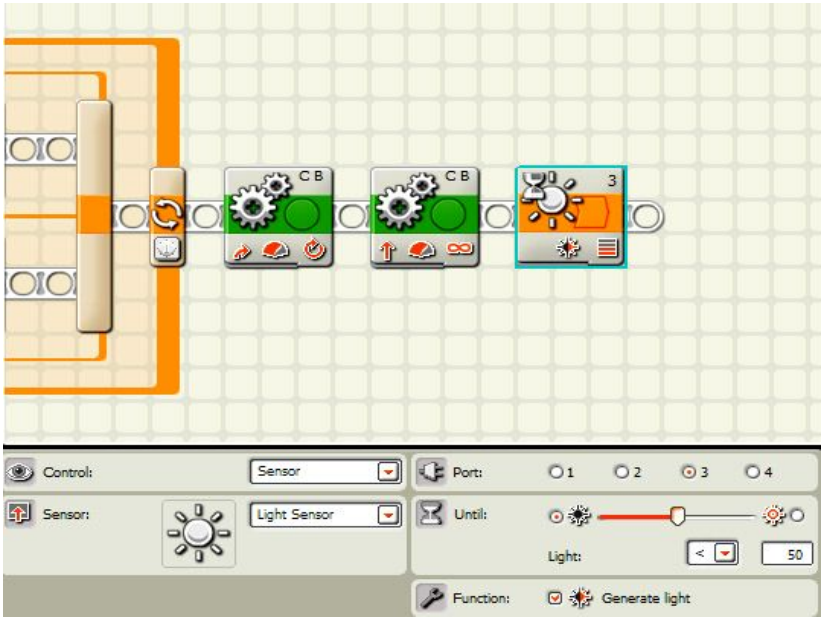


The move block set to unlimited keeps the robot moving until the next block stops it.

3. Put the cursor over the wait block and it opens up the different types of wait blocks. Choose the light.

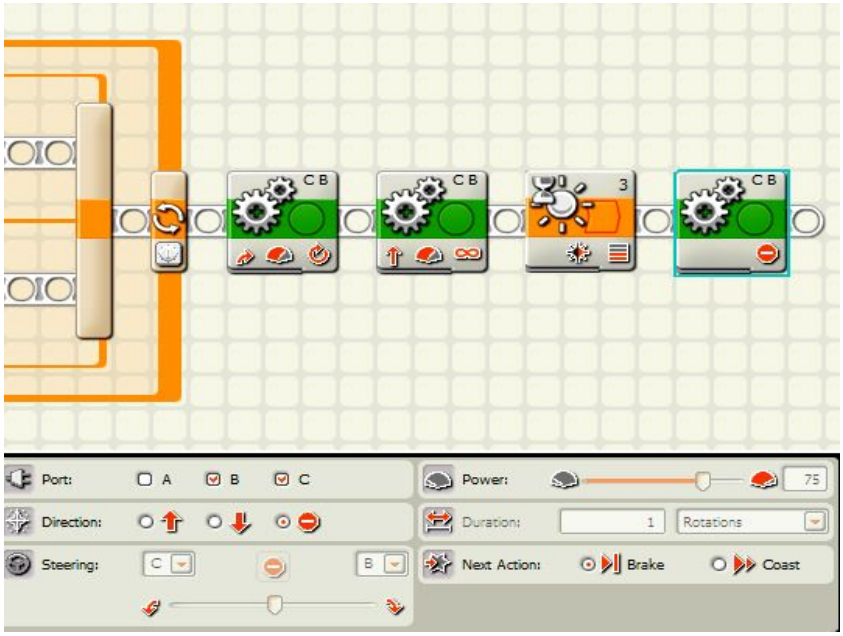


4. Place the light wait block on the bar and set it to less than and set the light level to 5 points lower than the white part of the practice pad.



This signals the robot when it crosses the black line of the big oval on the practice pad.

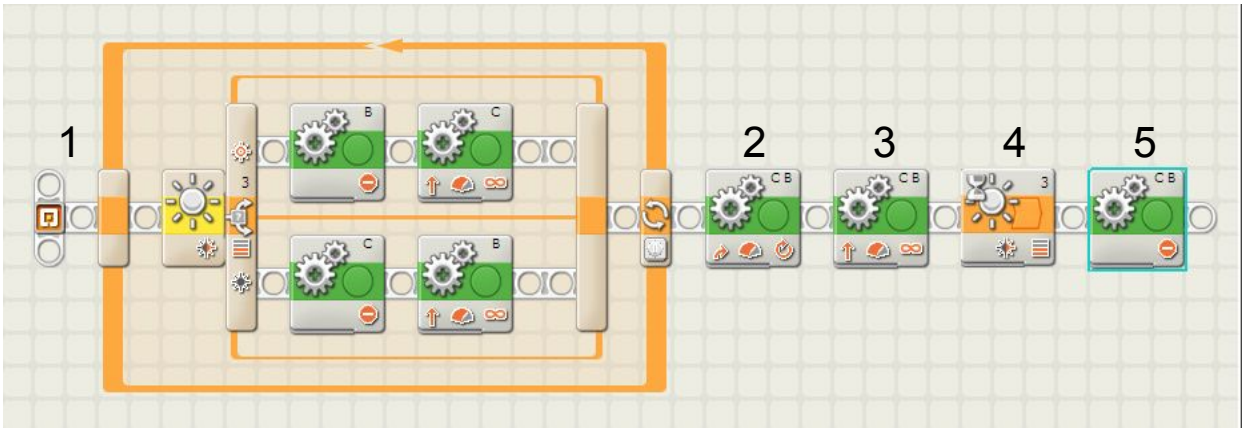
5. Place a move block on the bar and set it to stop.



This stops the robot after it crosses the dark line of the oval.

This is the whole program. It works this way:

1. The loop makes the robot follow the line for as long as the time is going.
2. When the time on the loop runs out, the move block turns the robot.
3. The next move block makes the robot keep going until the wait block is triggered.
4. The light block signals when the robot crosses the dark line of the oval.
5. The move block stops the robot.



Stop on Gray

Mission: the robot will stop on the right gray square when presented with 4 different gray squares laid out in a line in any order.

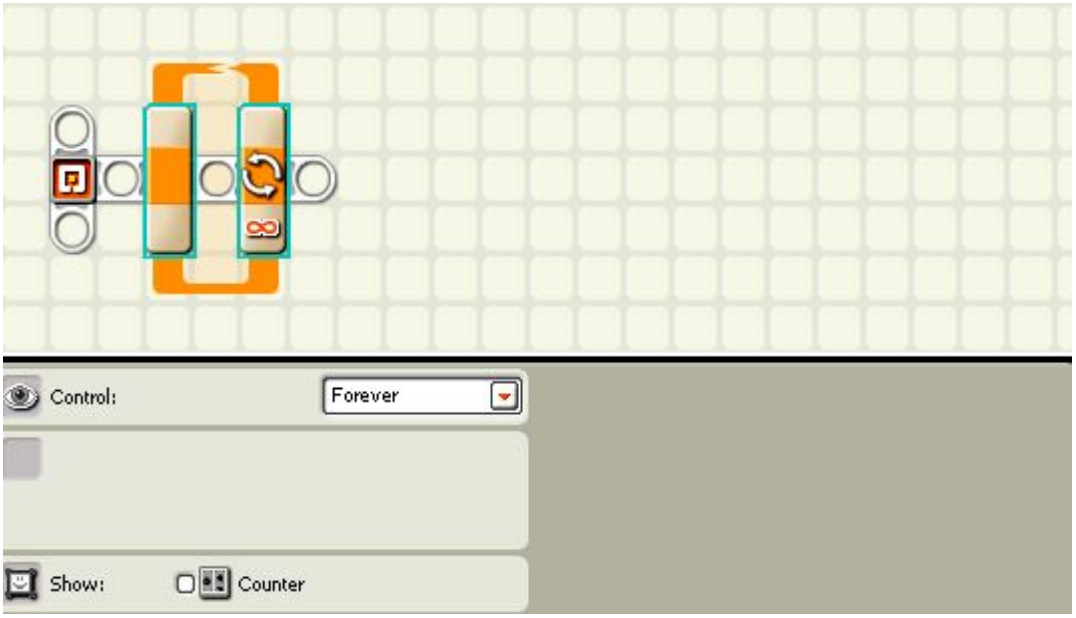
Equipment:
copies of the four gray squares following these instructions. (These gray rectangles are found after the directions. Copy them to use on the practice table. Put them under the clear plastic.)

Pict of field

Sensors:
light

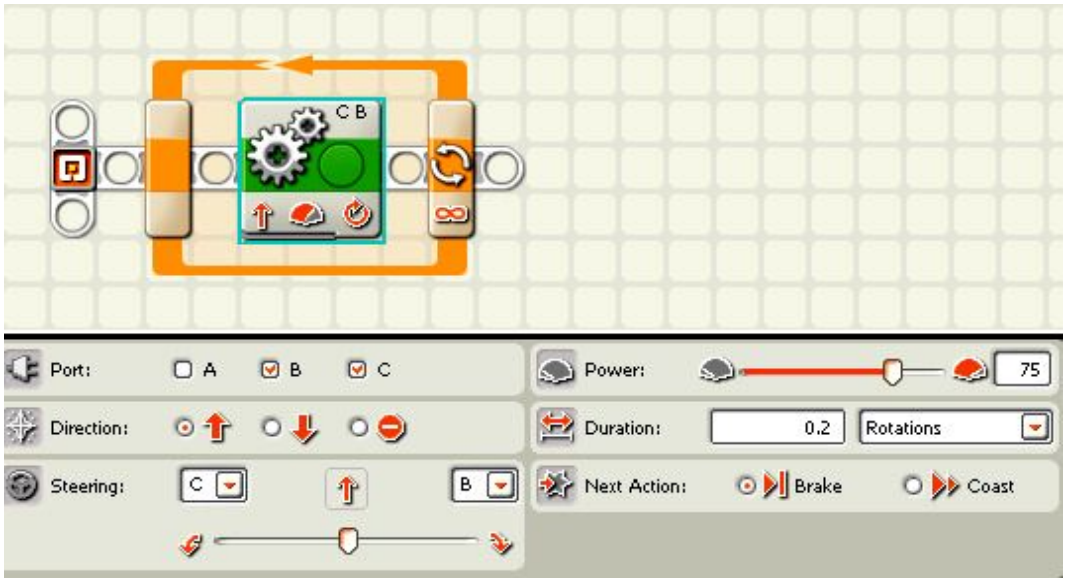
Directions:

1. Place a wait block on the program bar. Leave it set for Forever.



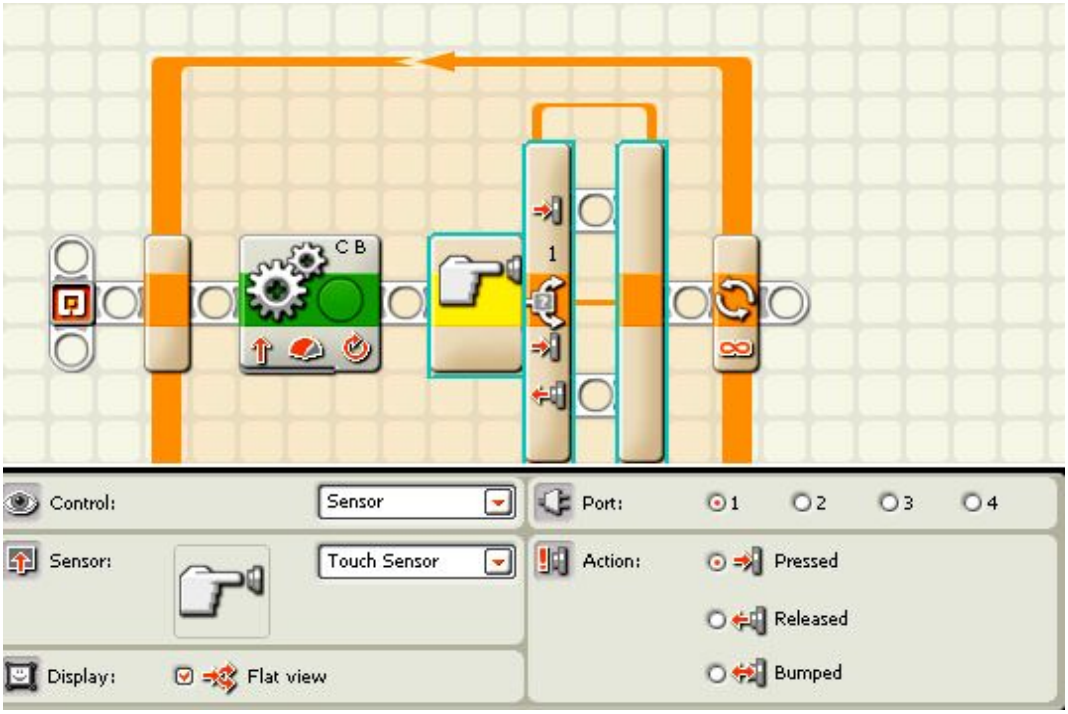
This makes the program run over and over until the program is stopped by pushing the dark gray button.

2. Place a move block inside it and set the rotations to 0.2.

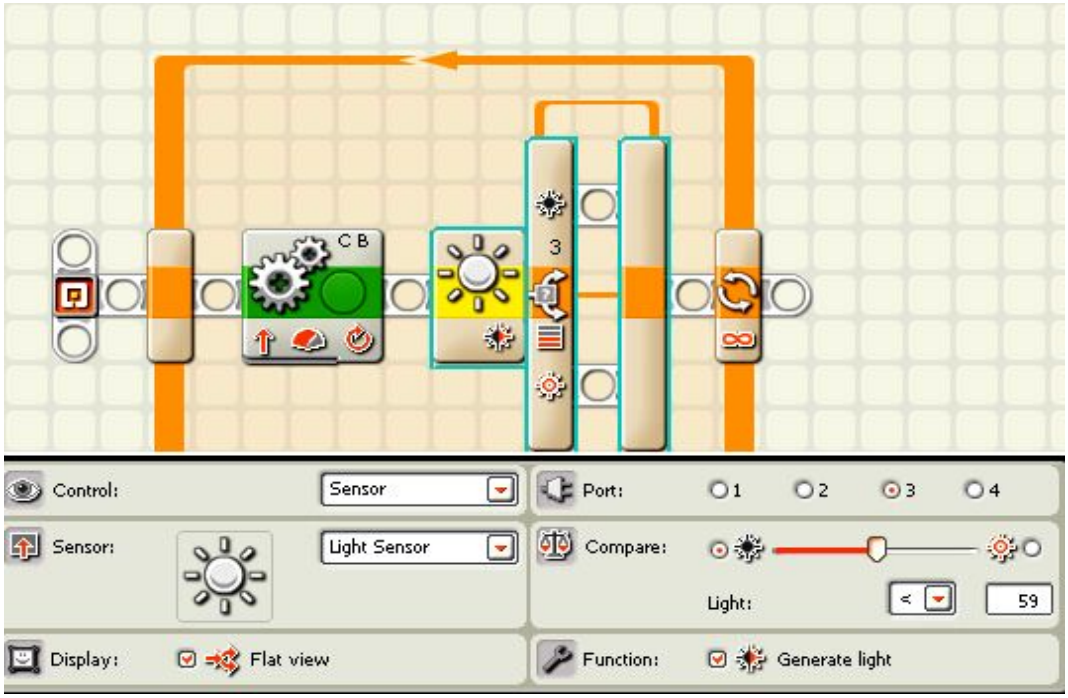


This gets the robot moving a short distance.

3. Place a switch block after the move block.

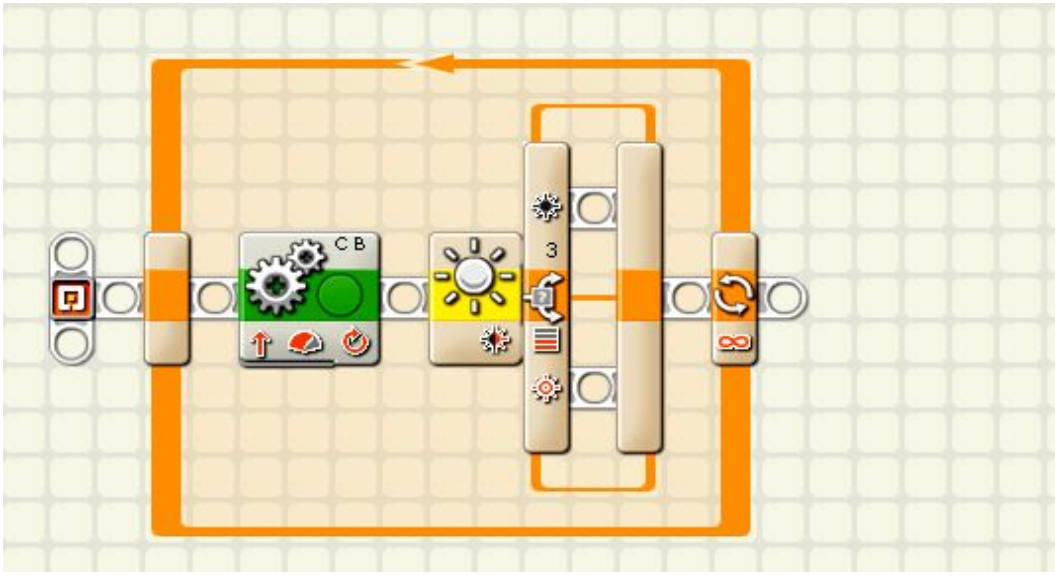


4. Change the touch switch to a light switch and set to a few points above the reading of the gray square you want to stop on. It is 59 in this case. Yours will be different.

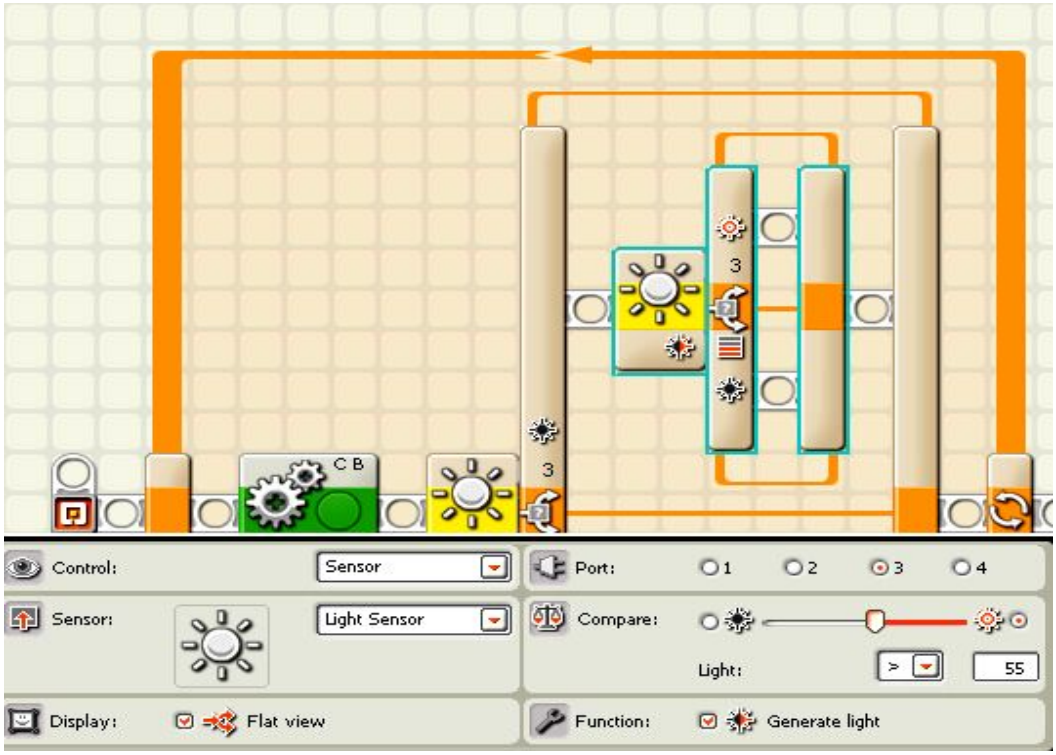


The light switch will make the robot ignore all gray rectangles that are brighter than 59.

5. The program so far will look like this.

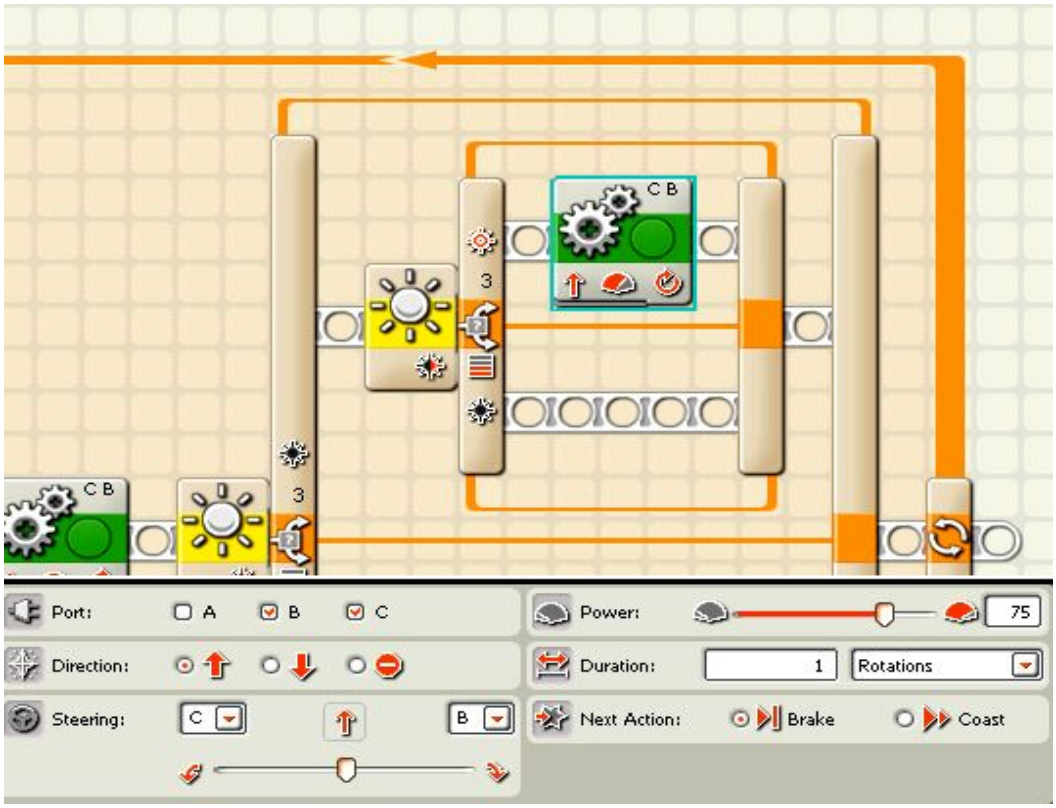


6. Place a switch inside the top bar. Set it a few points less than the reading of the gray square where you want the robot to stop. It is 55 points in this case. Yours will be different.

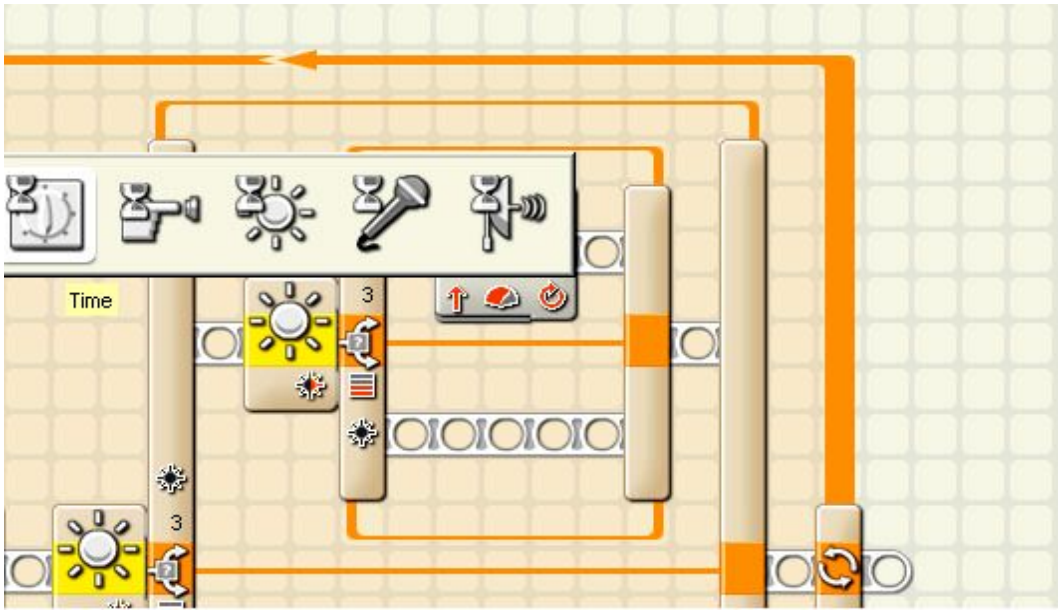


This switch tells the robot if the rectangle is just a few points less than the gray square you want. This eliminates the rectangles that are darker than the one you want.

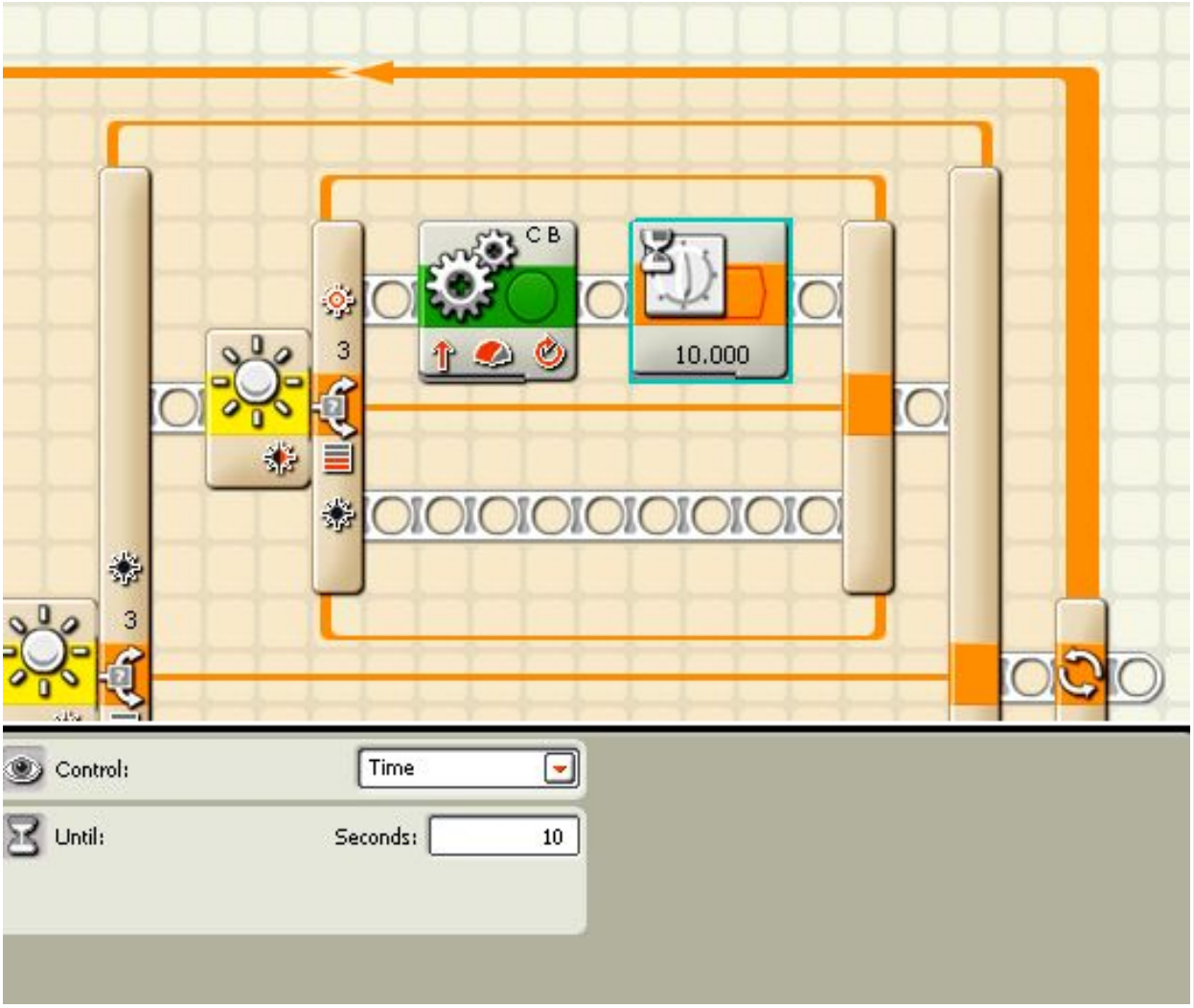
7. Place a move block and set it to 0.2 rotations.



8. Place your cursor over the wait block and a line of various wait blocks will appear. Choose the time block.

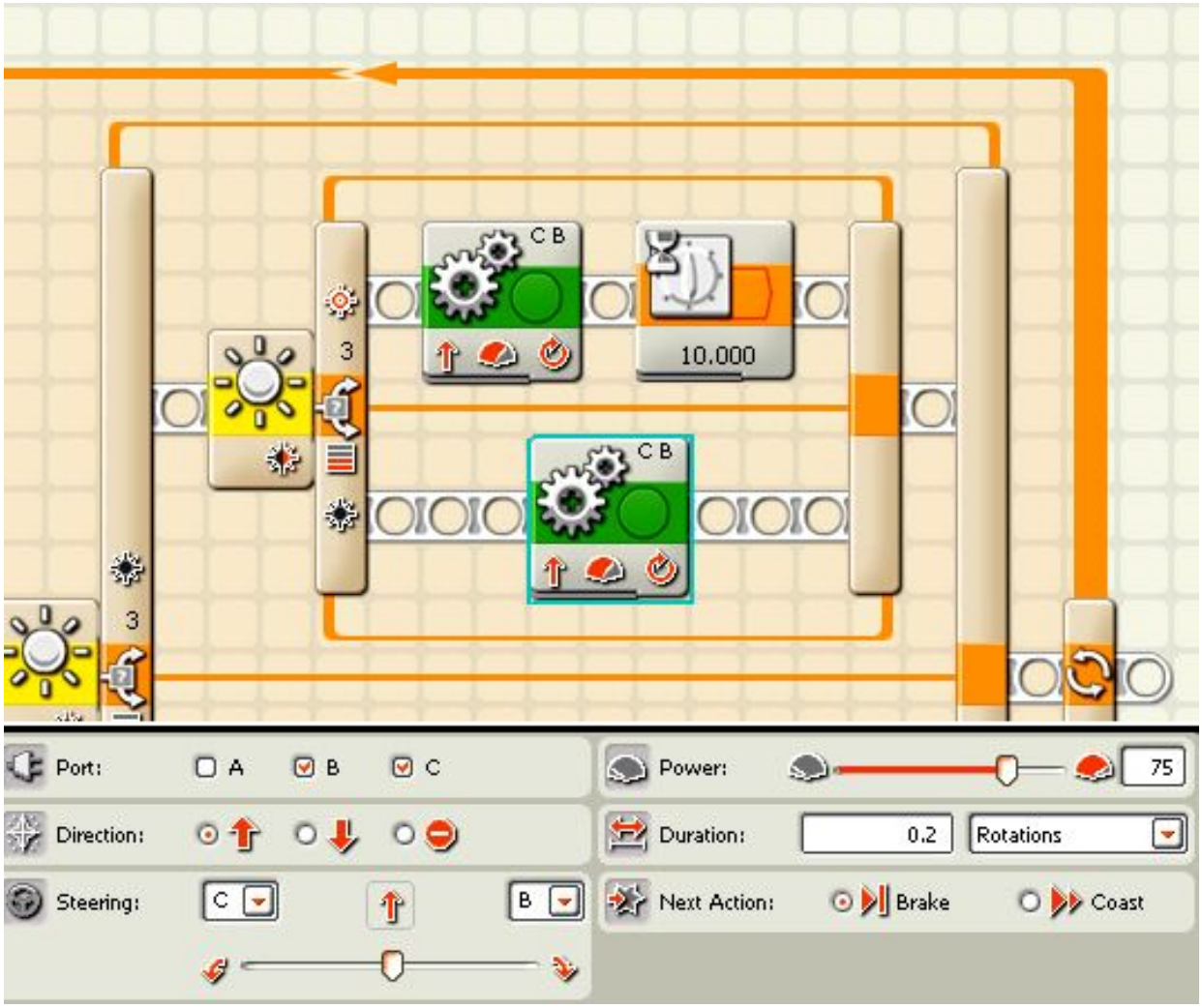


9. Place the time wait block. Set it to 10 seconds.



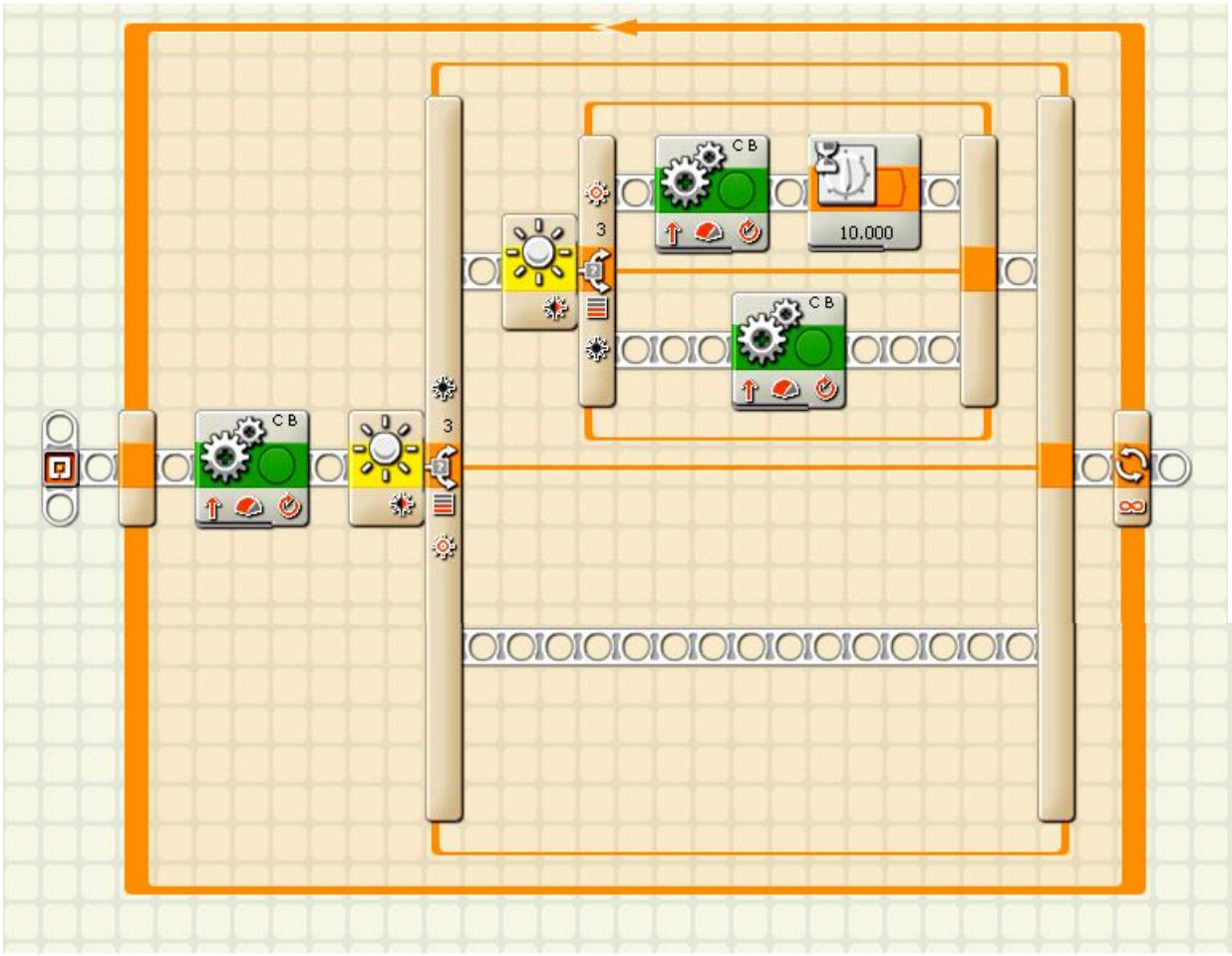
This makes the robot do nothing for 10 seconds which is long enough to grab it and stop the program.

10. Put another move block below on the lower bar of the inside switch and set it to 0.2 rotations also.

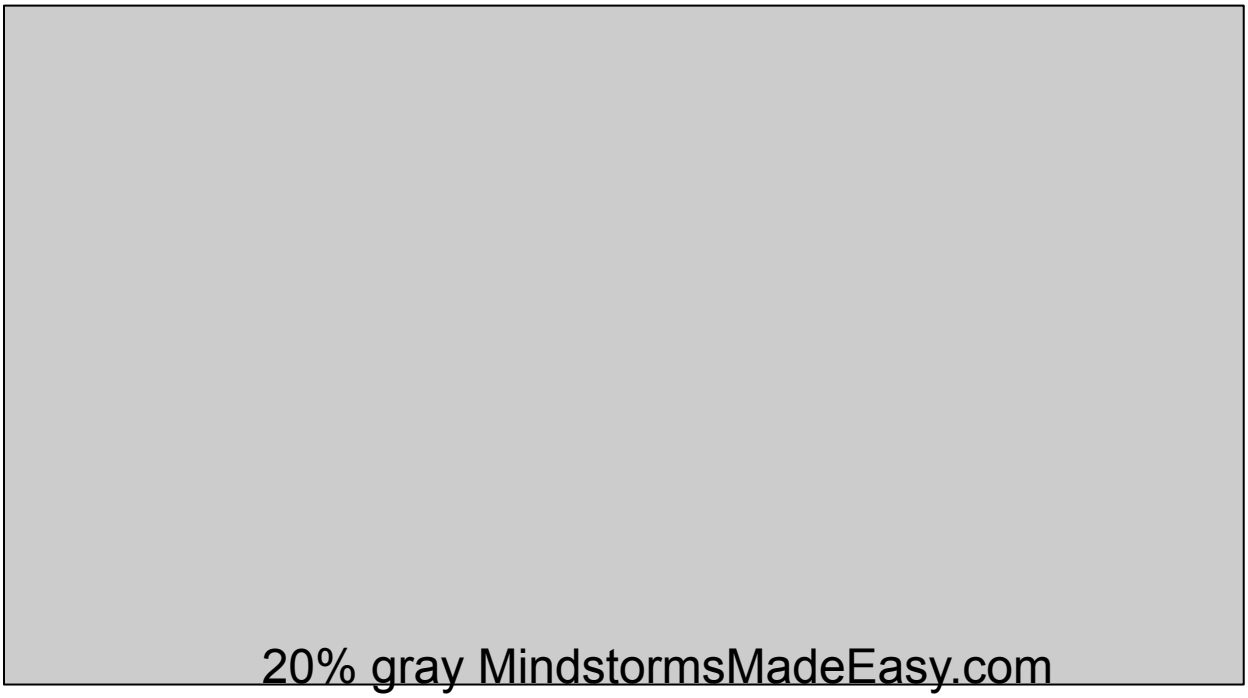


This keeps the robot going if the gray rectangle is not light enough to stop on here.

11. The whole program should look like this.



Cut out these rectangles into individual sections.



Cut out these rectangles into individual sections and have them laminated to help them last.

Mouse

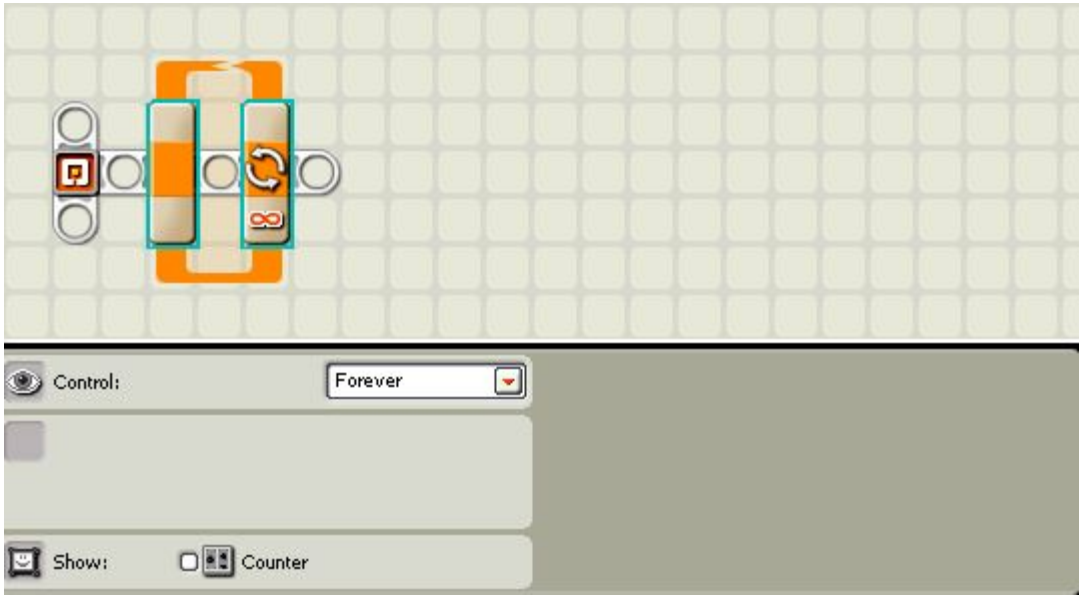
Mission:
The robot will move randomly until it finds a dark spot and will stop there.

Equipment:
a box cut in half diagonally

Sensors:
ultrasonic
light

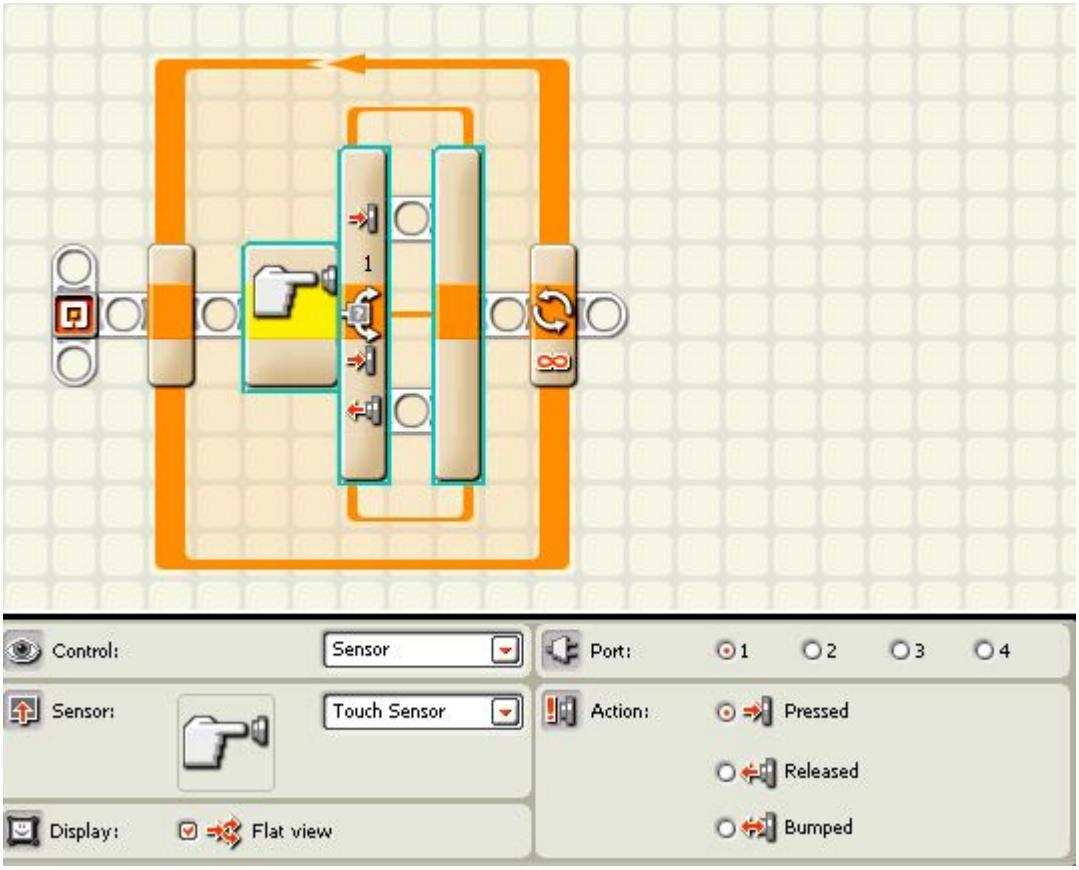
Directions:

1. Place a loop on the bar. Leave it set to forever.

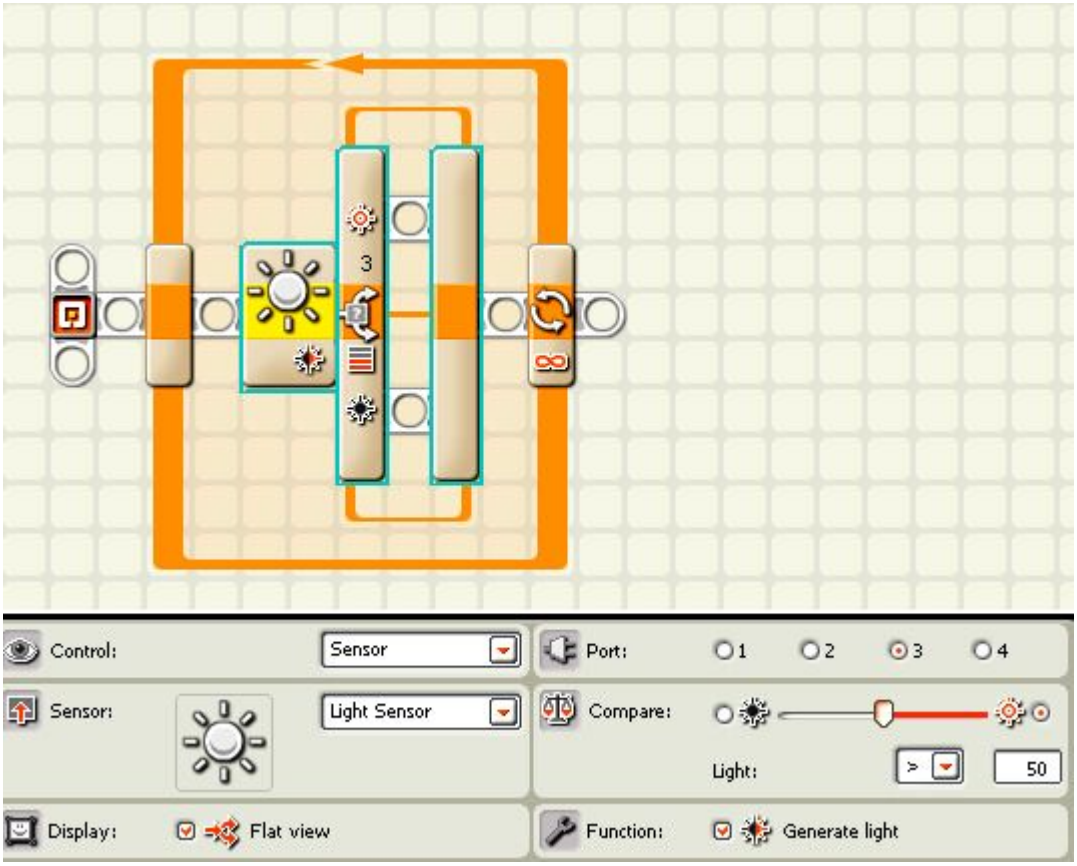


This will make the program repeat over and over until it is turned off.

2. Place a switch block inside the loop.



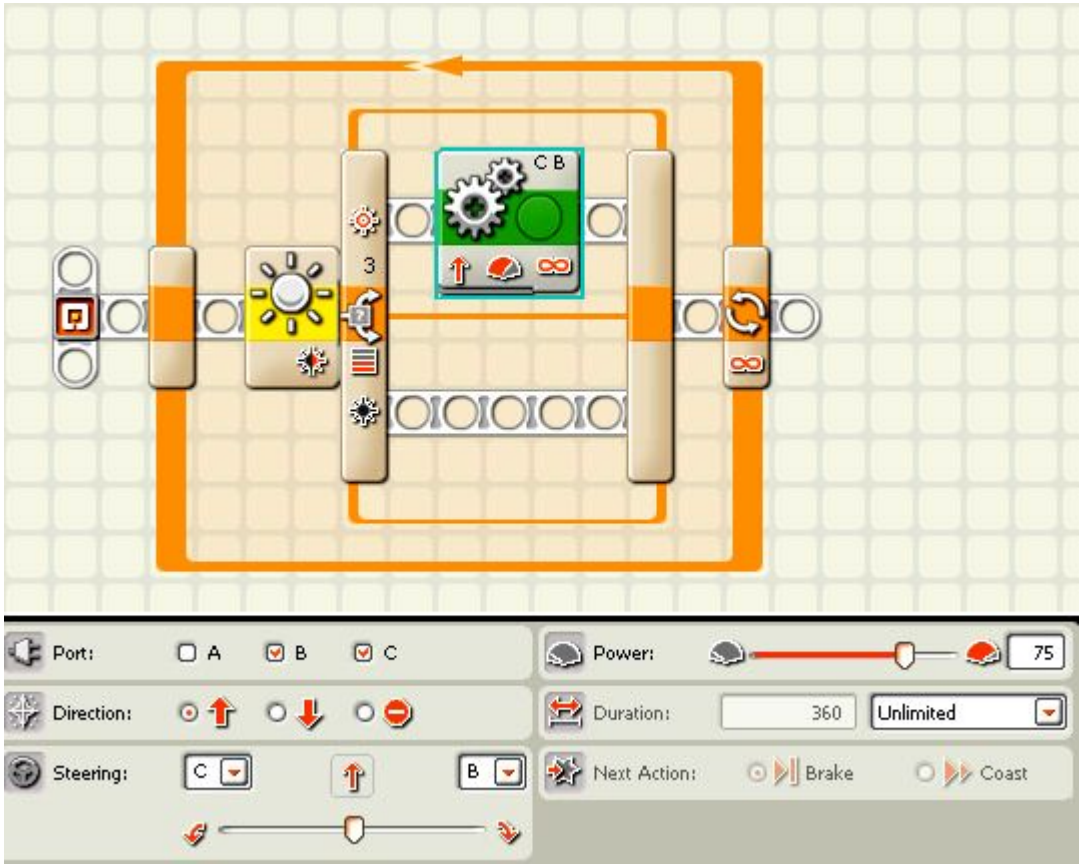
3. Change the touch switch block into a light switch block by using the drop down menu to the left of where it says Sensor in the bottom left.



This will make the program make a decision about how bright the light is above the robot. Set it 3 points less than the light reading outside the box on the practice pad.

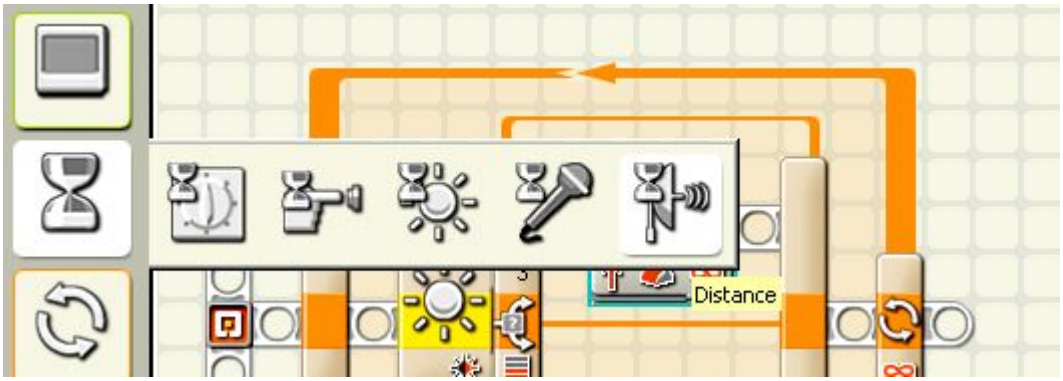
Be sure to take the reading without leaning over the robot and casting a shadow on it.

4. On the top line of the switch, set a move block and set it to unlimited.

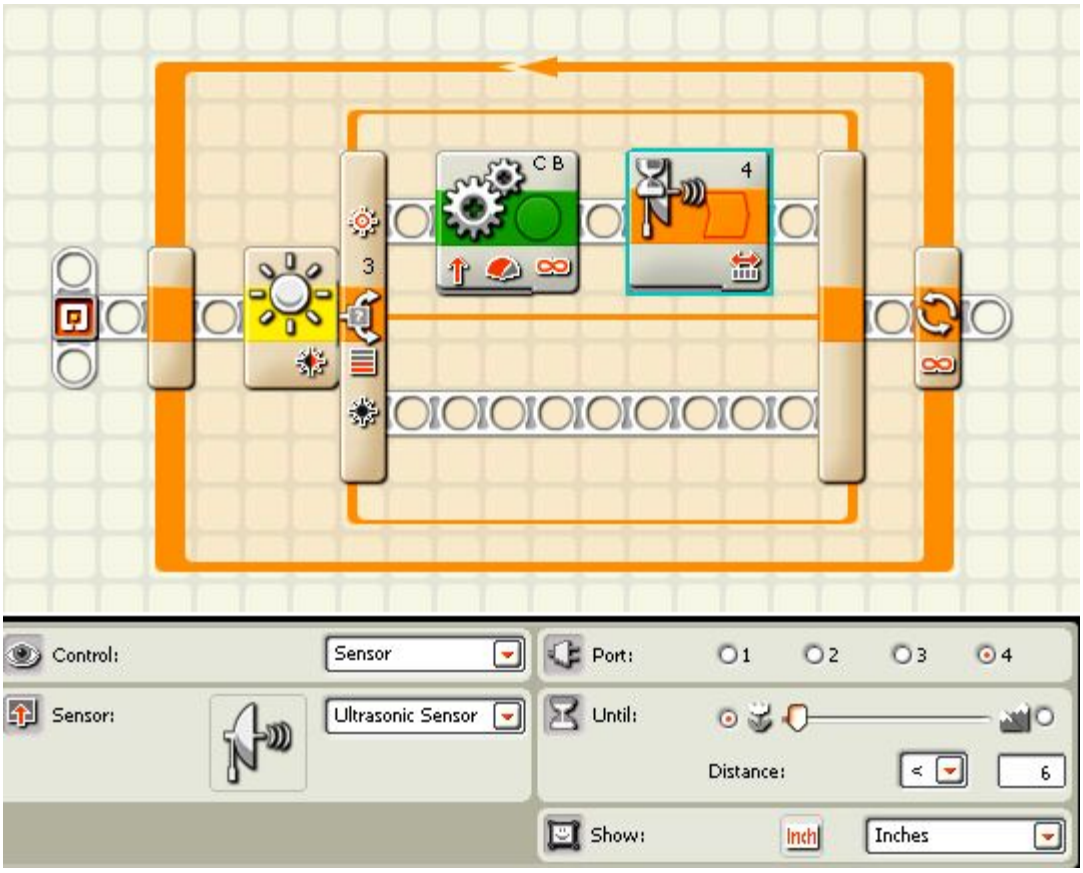


This will make the robot keep moving until the wait block you will put down next stops it.

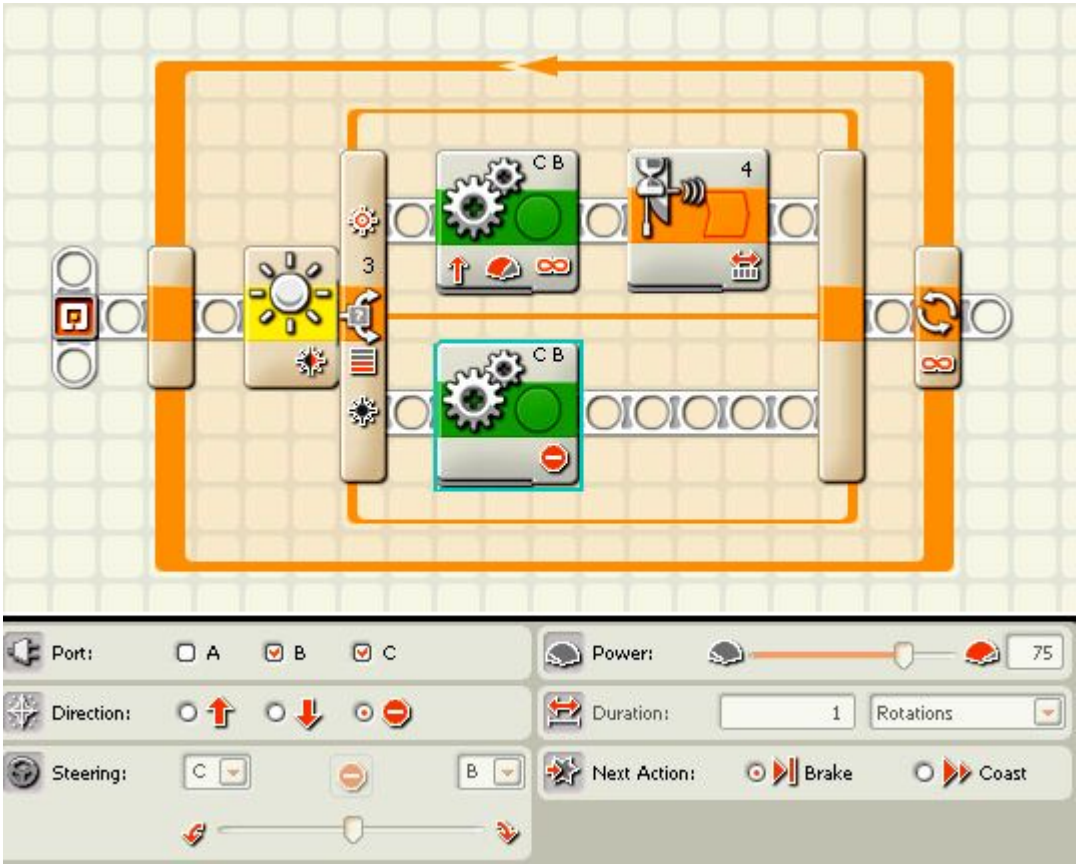
5. Put your cursor over the wait block and a menu will open up. Pick the distance block.



6. Place the distance block to the right of the move block and set it to less than 6 inches.



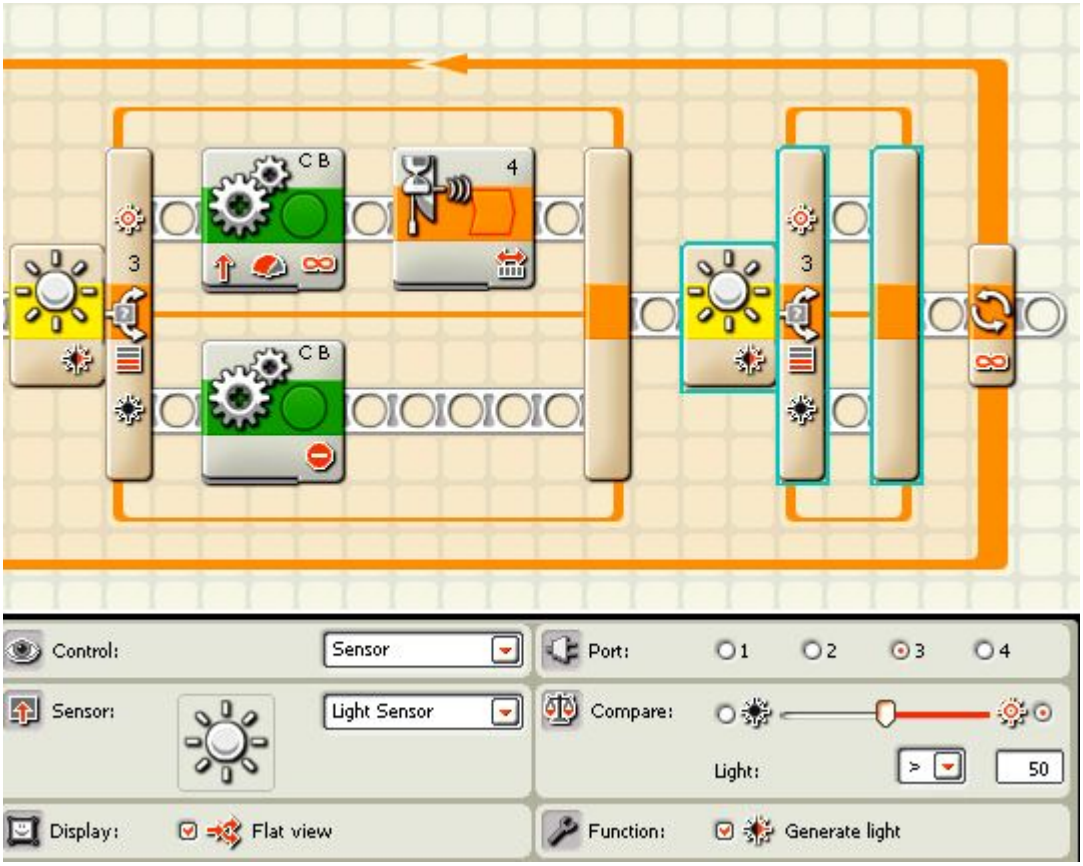
7. Place a move block on the lower line and set it to stop.



If the light switch decides that the light is too low so the robot is in the shade, it will use the bottom line of the program and make the robot stop. This will continue until the program is stopped or the box is moved away and lets the light shine on the robot.

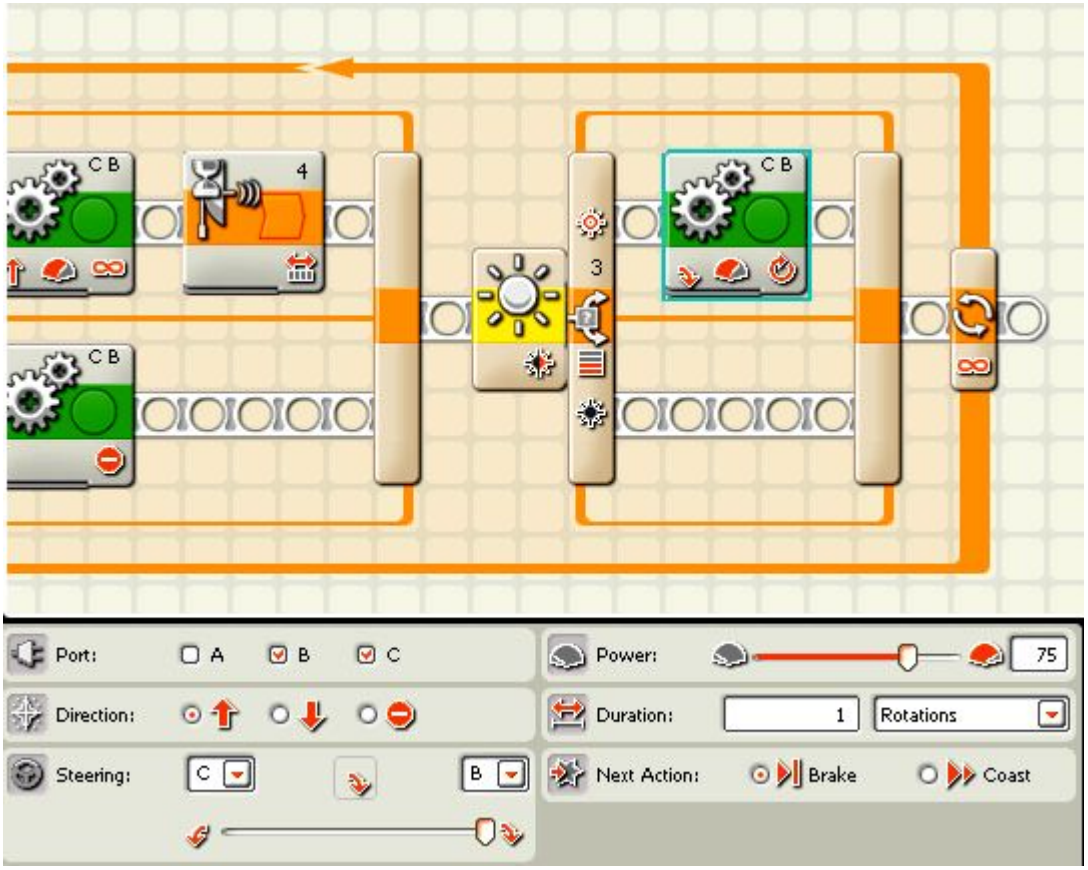
To review, this part of the program has a loop to repeat the action, inside the loop there is a switch to measure how bright the light is. If it is bright enough, the program decides the robot is in the light, takes the top line of the program. The robot moves until it comes close to a wall. If the light is not bright enough, the robot decides that it is in the dark and it takes the bottom line of the program and the robot stops.

8. Place a switch on the bar after the first switch and change it to a light switch.



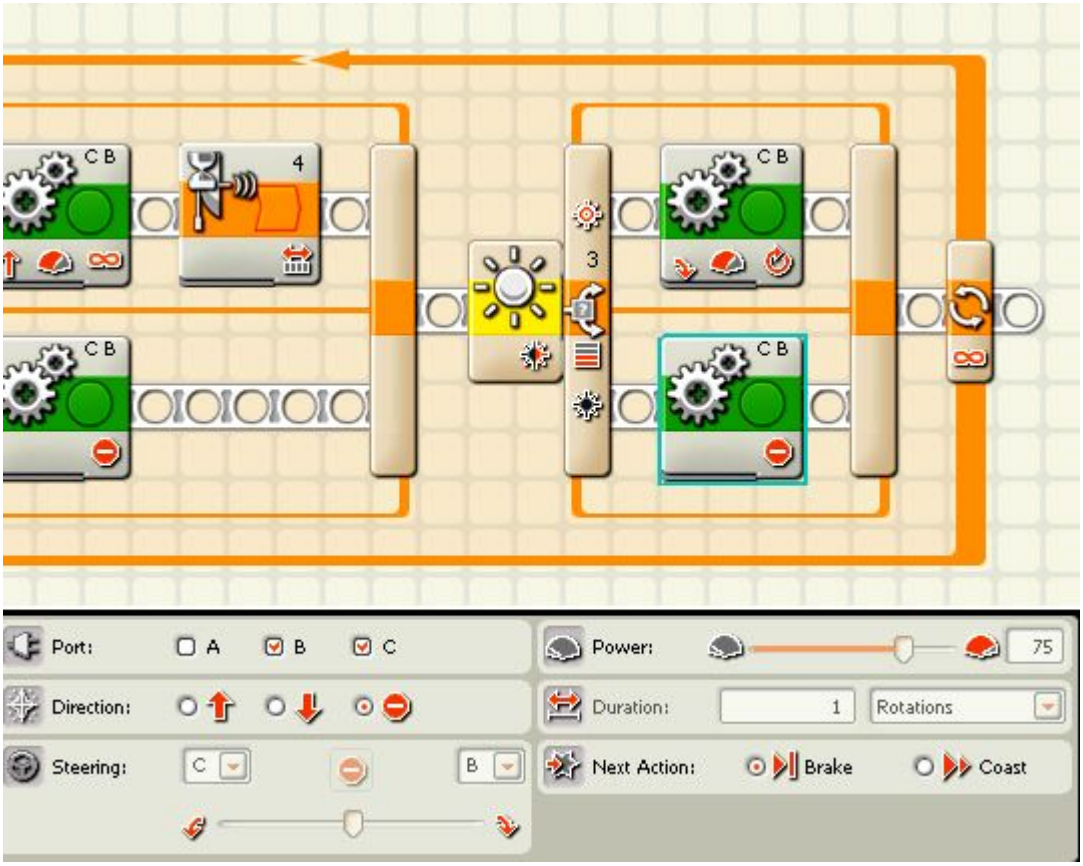
This will test the light level again to see if the robot is in the dark now that it has moved close to a wall.

9. Place a move block on the top line and leave the rotations at 1. Slide the steering slider all the way to the side.



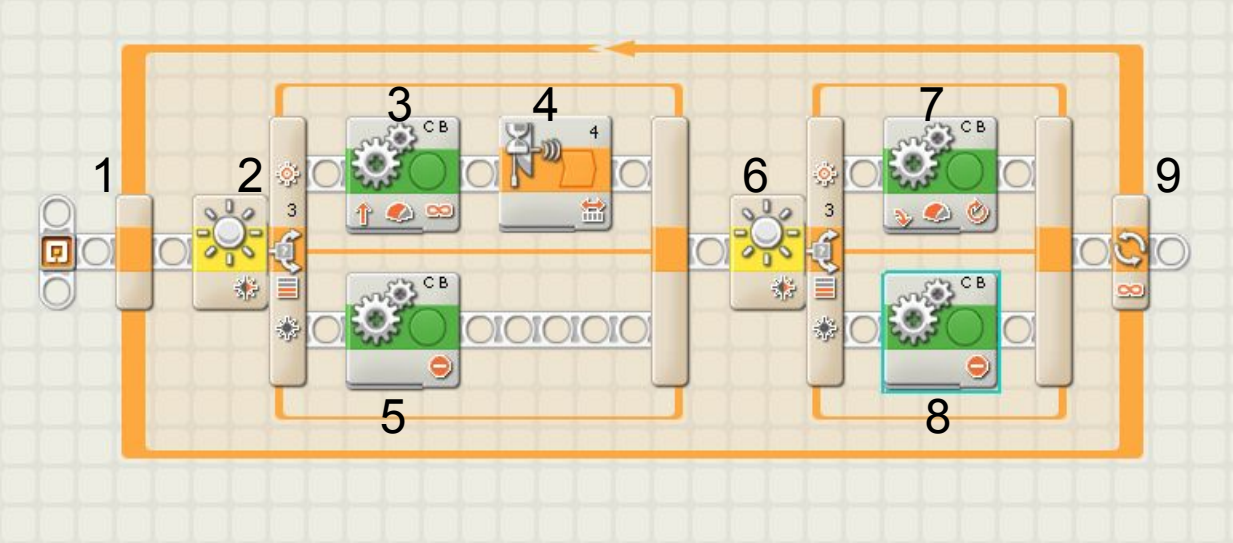
The switch decides if the light is bright enough. If it is bright enough, the program follows the top line and turns the robot so that it will be ready to move to another wall when the program repeats. It will repeat because of the loop.

10. Place a move block on the bottom line and set it to stop.



If the switch block decides the brightness is dark enough, it assumes the robot is in the shade and follows the bottom line. The move block is set to stop so the robot stops.

11. This is the whole program.



1. The loop will make the program repeat endlessly until the program is ended.
2. The light switch will test the light to see how bright it is. This will decide if the robot is in the light or in the shade from the box.
3. If the switch decides it is in the light, it takes the top line and has the robot move until the next block stops it.
4. The distance block uses an ultrasonic sensor to check how close to a wall the robot has come. When it is less than six inches away, it stops the move block and the program moves to the next step which is the next light switch.
5. If the light switch decides that the robot is in the dark, it follows the bottom line. The move block is set to stop the robot so it stops. It will stay stopped as long as the robot stays in the dark or until the program is stopped.
6. The next light block tests the brightness again. If it decides the brightness is high it takes the top line. If it decides the brightness it not high, it takes the bottom line.
7. If the robot is in the light, it turns to get ready to move again.
8. If the robot is in the dark, it stops. It will stay stopped as long as the robot stays in the dark or until the program is stopped.
9. The loop causes the program to return to the beginning and start the whole process over.

Ultra Hit Again

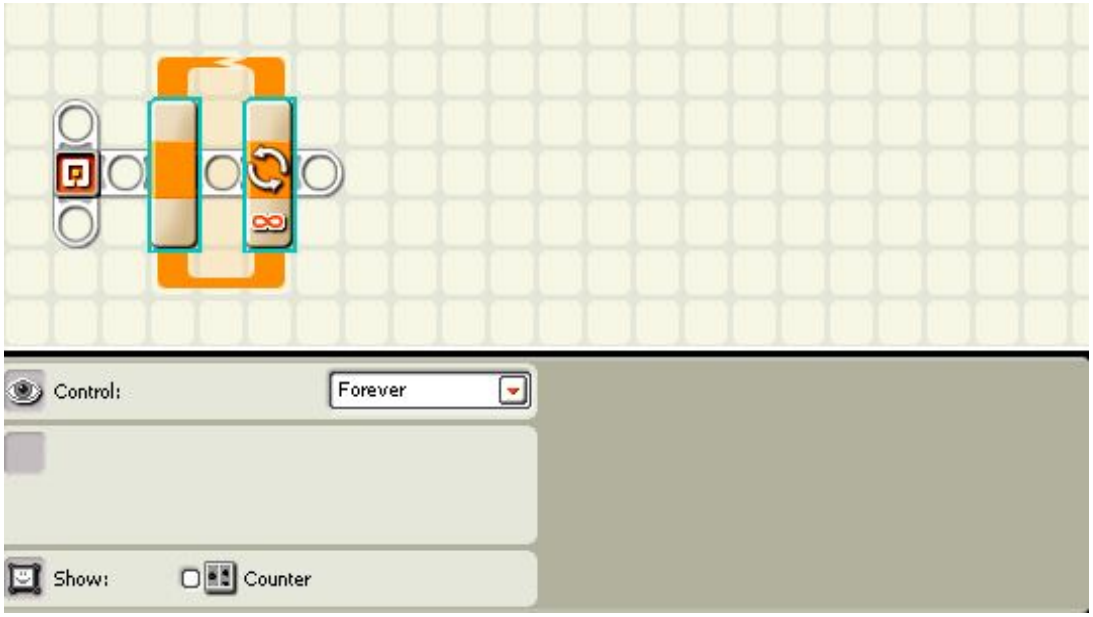
Mission: the robot will move forward until it touches the wall, it will then back up until the ultrasonic sensor senses that it is one foot away from the other wall. It will do it over and over again.

Equipment:
none

Sensors:
touch
ultrasonic

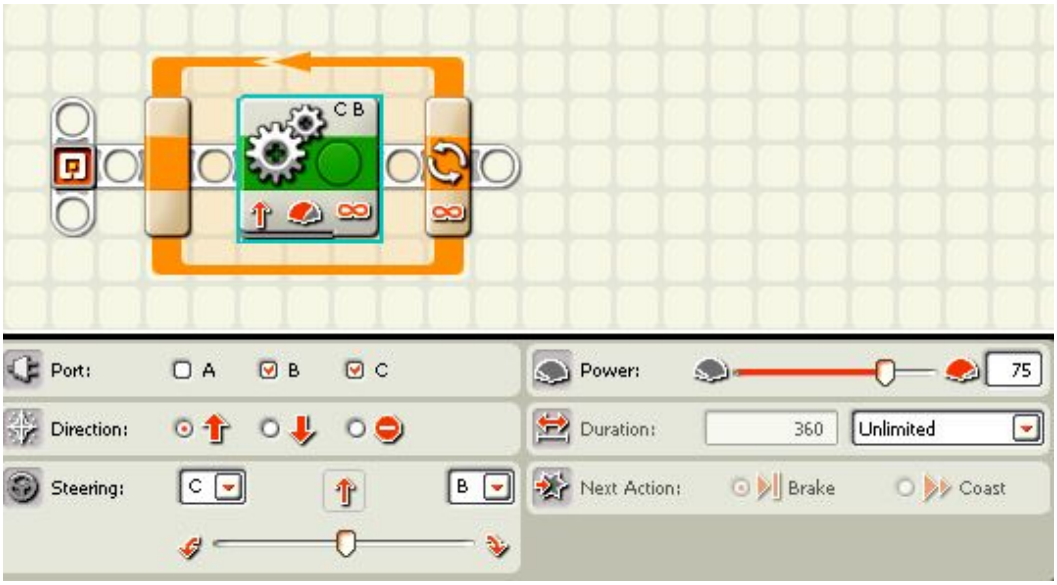
Directions:

1. Place a loop on the bar and leave it at forever.



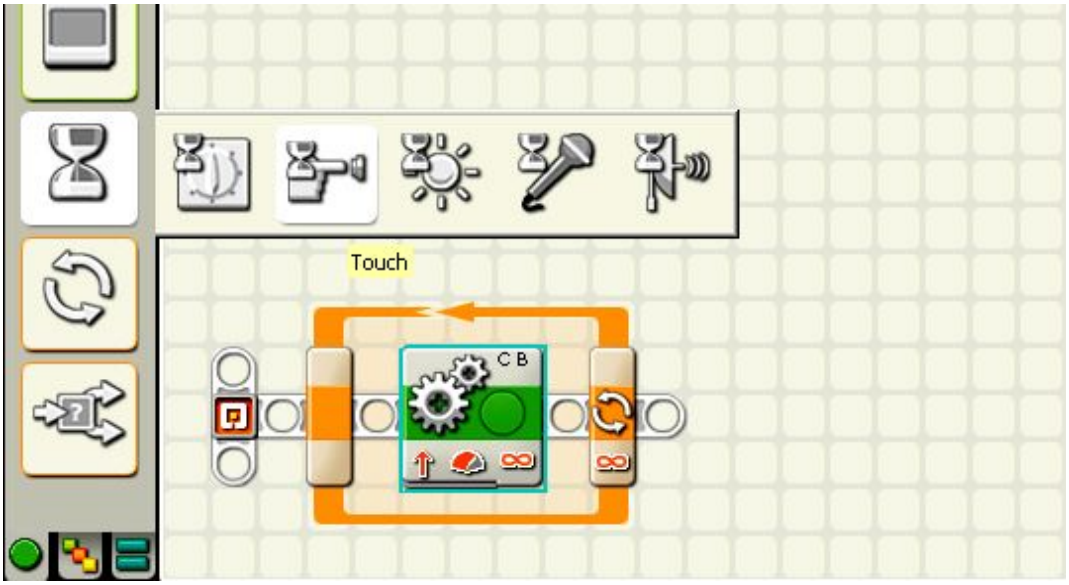
This will make the loop repeat over and over.

2. Place a move block into the loop and set it at unlimited.

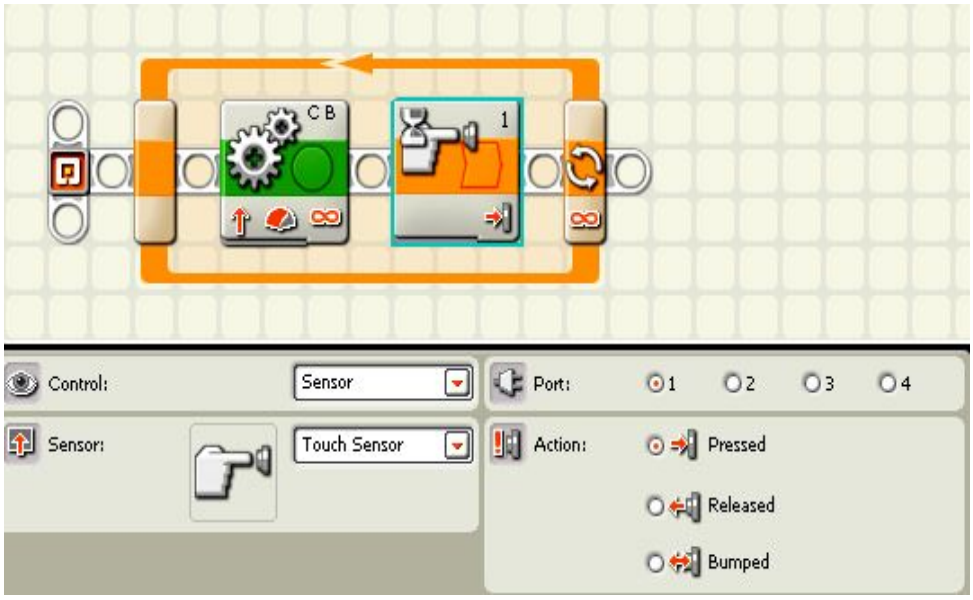


This will make the robot move for an unlimited distance until the touch wait block tells the program to move on to the next block.

3. Move the cursor over the wait block and a series of different types of blocks will open up. Pick the touch.

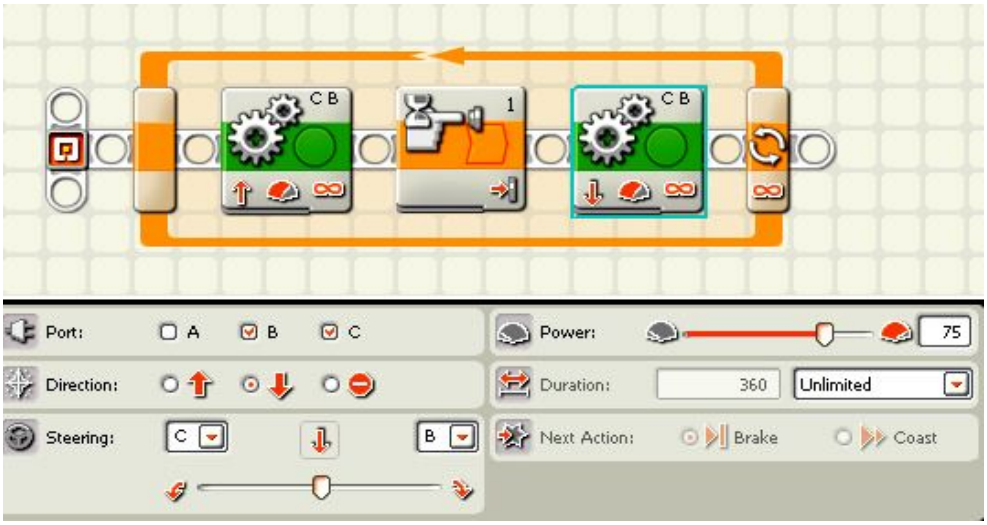


4. Place a touch wait block on the bar and leave it at pressed.



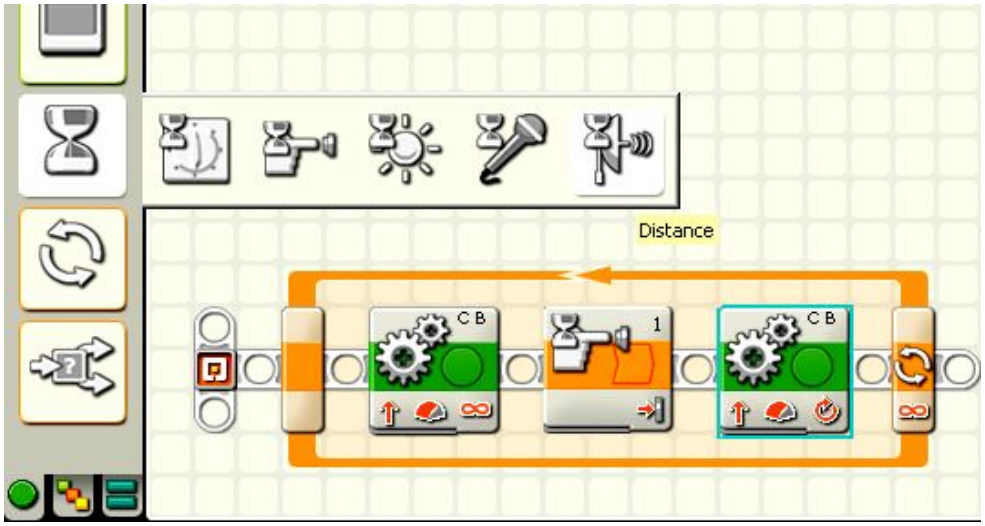
This makes the robot stop when the touch sensor is pushed and moves the program to the next block.

5. Place a move block on the bar and set it for reverse and unlimited.



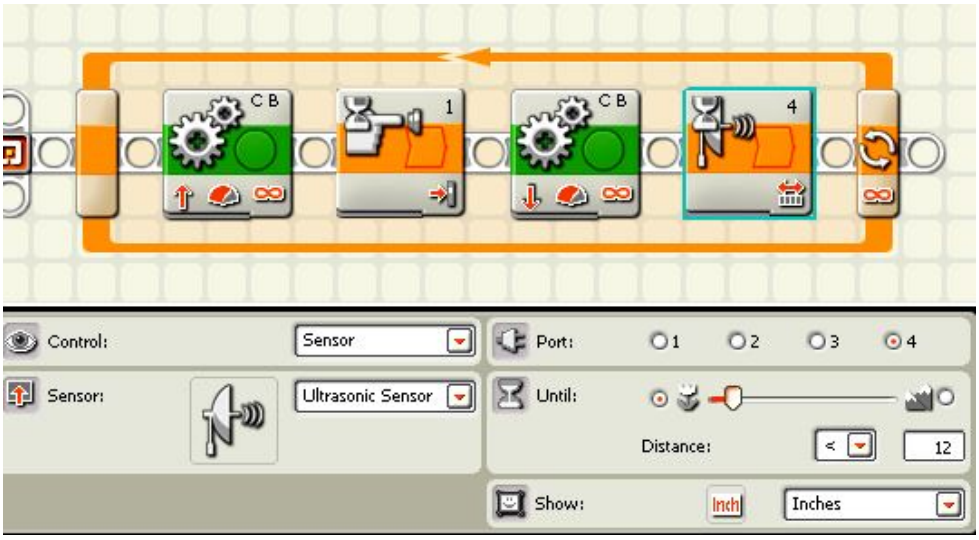
This makes the robot back up until it is told do do something else.

6. Place your cursor over the wait block. A menu will open up. Pick the distance block.



The distance block works with the ultrasonic sensor to judge distance by sending out an ultrasonic ping and measuring how fast it comes back and then calculates the distance.

7. Place an distance wait block and set it for less than a foot.



This will make the robot end the reverse move block when the robot moves to a distance less than one foot from the back wall. Then the loop will bring the program back to the beginning and the robot will move forward again until it hits a wall, back up until it gets less than one foot away and start the whole process over and over.

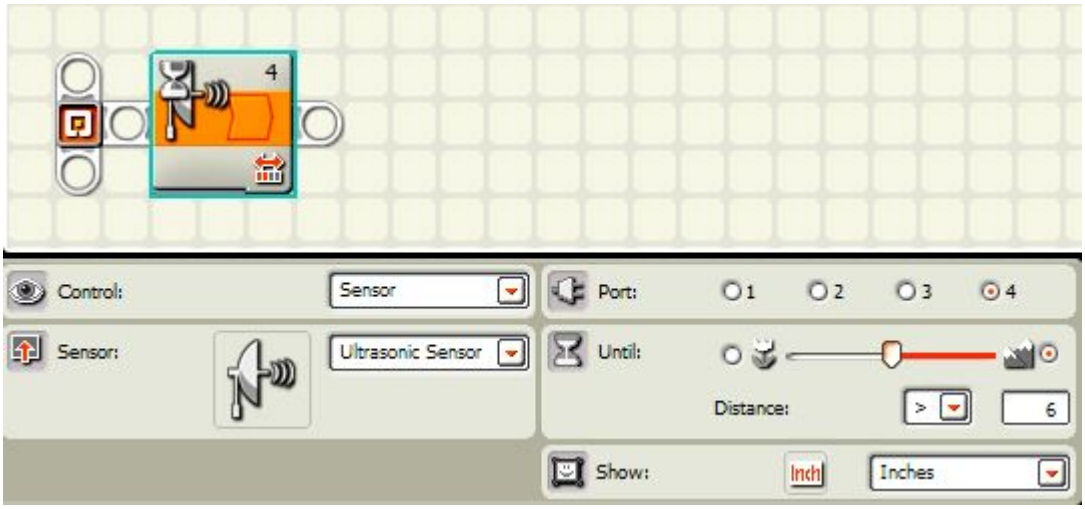
Contest Start

The following are games that let the students compete to complete a mission. It is good practice to do this sometimes when the students are getting tired of doing the same thing and need a little excitement. Rather than giving them extra credit, they can be rewarded for doing well with the winning students' name on the board or giving them a piece of candy or something along those lines.

Starting the Race

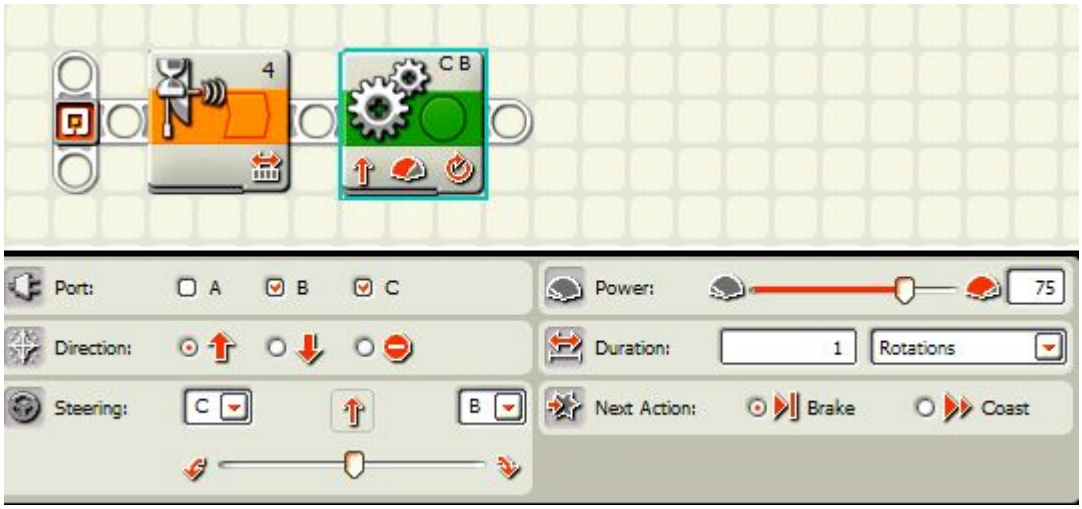
With any contest, it is important to have both robots starting at the same time. Each person could just push the button to start, but here is a more impartial way to do it. It is a short piece of code that allows one person to start both robots. The robots are held in place by a distance wait block that won't move to the move block until a block or even a student's hand is moved out of the way.

1. Place a move block on the bar and set it to less than and to 6 inches.



This will make the robot wait until the block in front of it is moved. The program will wait until the distance in front of the robot is greater than 6 inches.

2. Place a move block on the line.



The move block will allow the robot move forward to start the contest.

To review, the distance block will make the robot wait until the block in front of the ultrasonic sensor is moved. Once it is moved, the robot can move forward and start the contest.

You will then need to add the various blocks to make the robot complete the contest, and, with some luck, will be the winning robot in the contest.

Contests

One of the great ways to promote students to learn programming and also to reward the students who are working hard and learning how to program is to hold contests one in a while in your class.

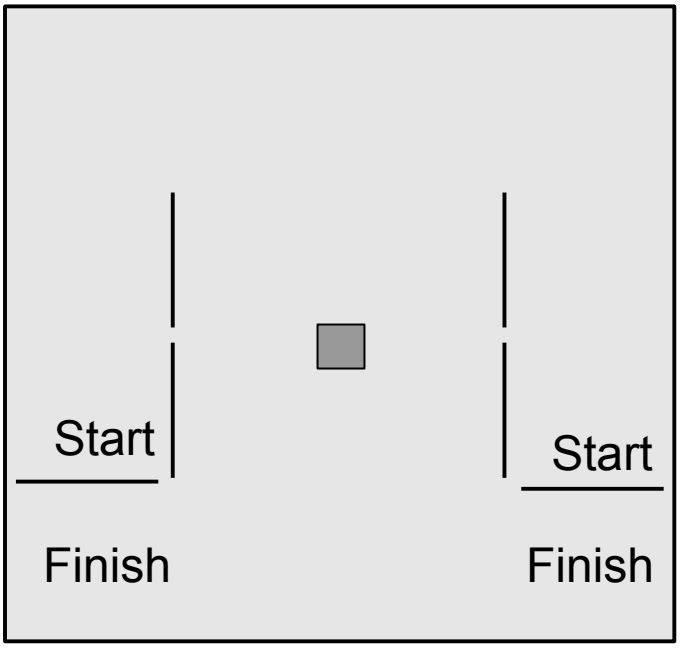
How to do this.

- Decide on a challenge. It can be one of the assignments in this book.
- Divide into teams. If you have several students per robot, just make each robot team a team.
- Time how long it takes to compete each challenge. Not only does the team need to do the assignment, but the one who does it the fastest wins.
- No touching the robot once it starts. The students can't nudge the robot if it is going in the wrong direction or if it didn't turn correctly.
- Remove knocked down pieces and reposition pieces that get moved out of place.
- Keep track of the times of each team.
- Have some kind of reward for those who do well. It doesn't need to be extra points. Usually the students who are going to win are the A students anyway. Food is good, but some parents don't appreciate the extra sugar. certificates of achievement, medallions, or small trophies are great. Don't forget the power of pictures. Use a digital camera or your cell phone to take a picture and put it on the wall of the classroom, or, better yet, in the hallway. This rewards the students with some amount of fame and also helps advertise your program. Make your class fun and cool and you will have students begging their councilors to sign up for it. That will help keep your class from being cut during budget cuts and pressures to improve test scores.

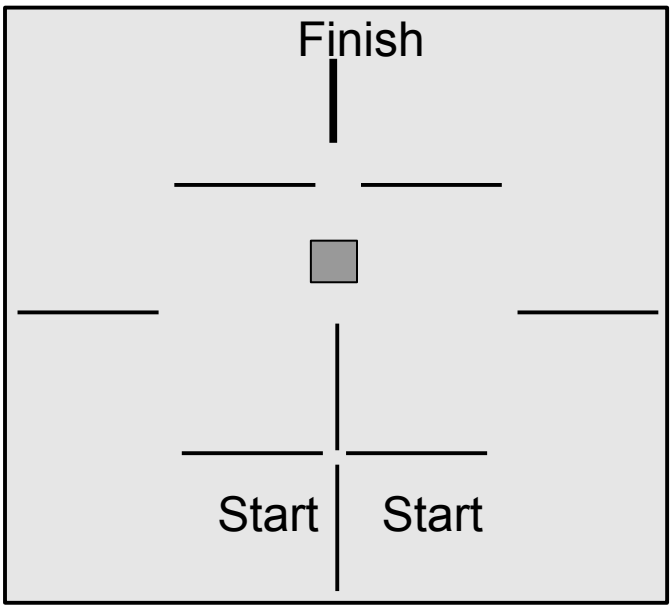
Steal the Flag

Mission: the robot will compete against another robot to capture a block that stands as the flag and get it to that team's home base.

#1



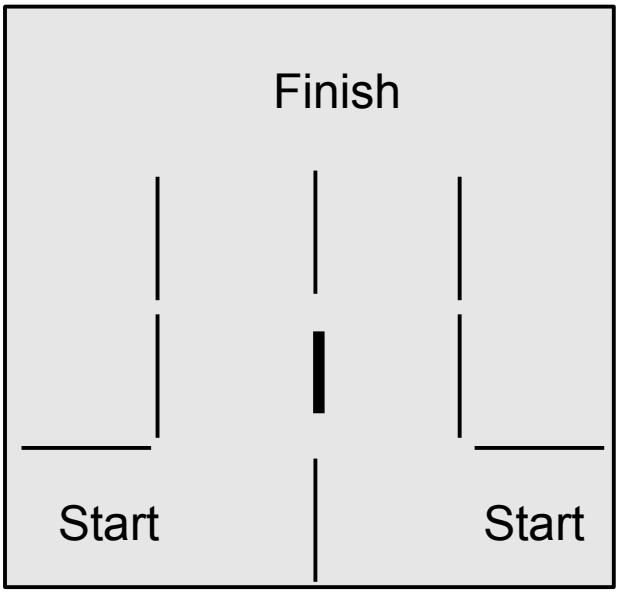
#2



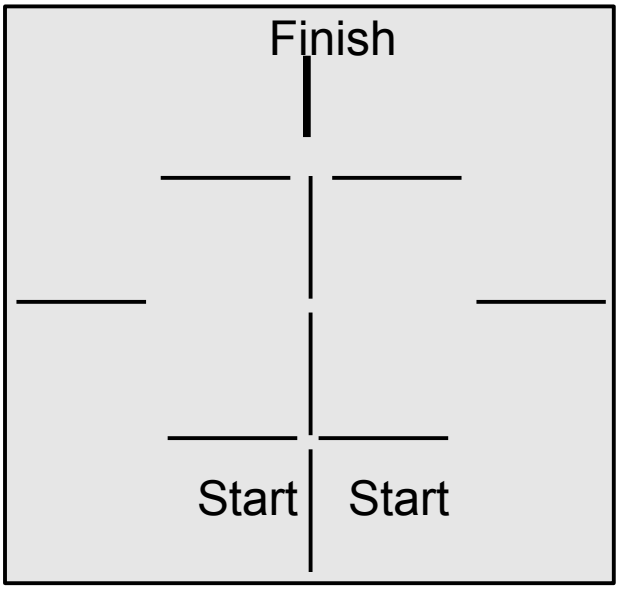
Knock First

Mission: the robot will compete against another robot to race around a short maze and knock over the block first.

#1



#2



Back Shimmy

Mission:
The robot will back up to a wall going in reverse but not at a 90 degree angle and by alternating between the left and right wheels, it will adjust itself so that it is now pointing at a 90 degree angle to the wall.

Note: This is a great way to get the robot properly oriented after it has traveled a long distance or made several turns since the robot has the potential to get slightly out of alignment with the walls.

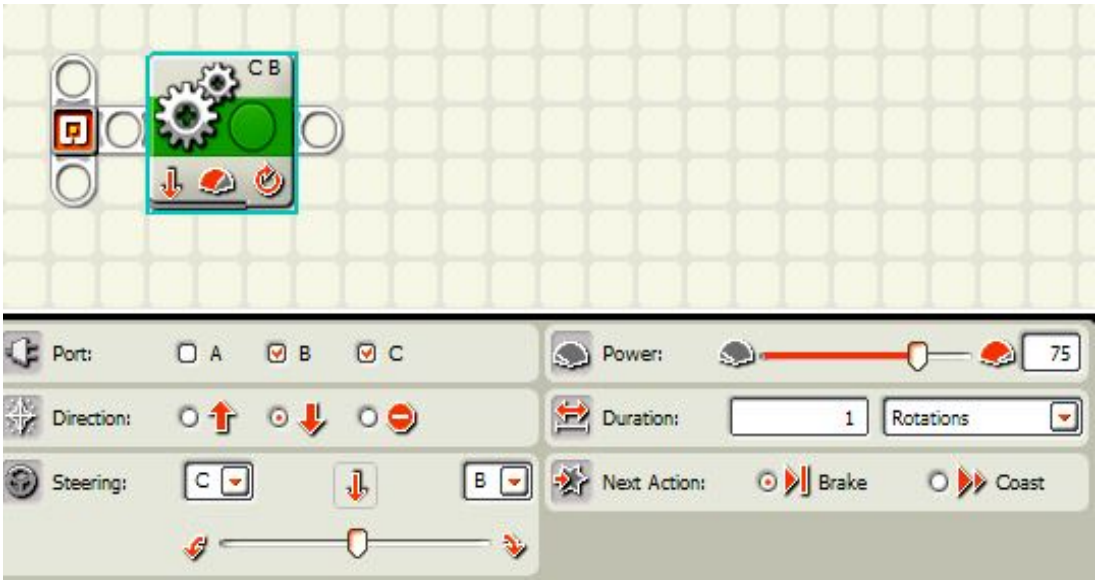
You will not need something as involved as this all the time. Sometimes, the robot just needs to bump up against the wall to orient itself if it is not too far from being at a right angle to the wall.

Equipment: The robot needs a flat bumper on the back to use to back up against the wall.

Sensors: none

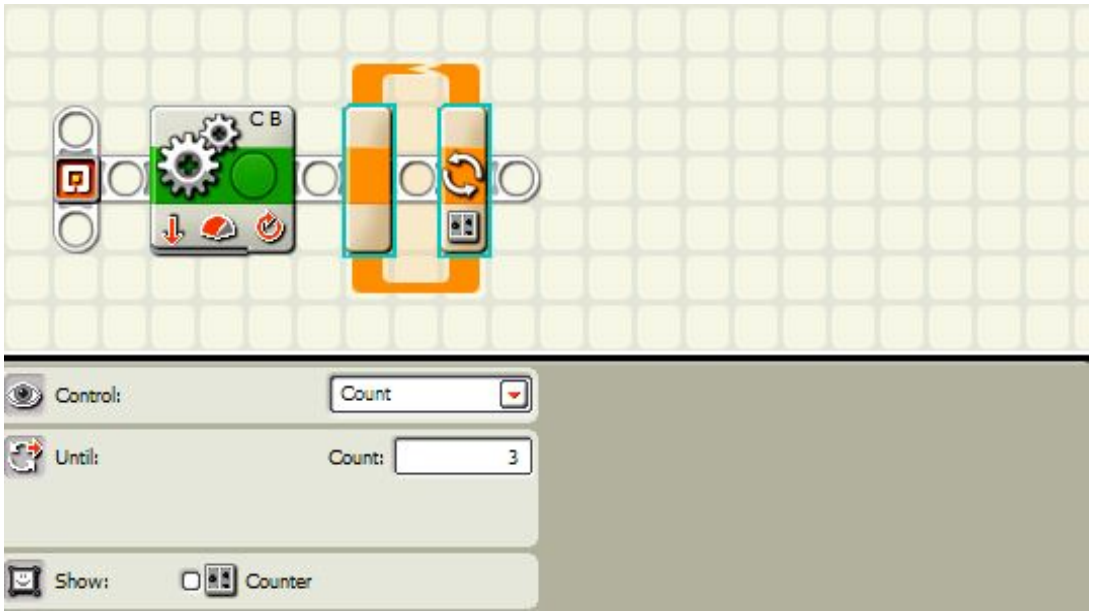
Directions:

1. Place a move block on the bar and set it for reverse and leave it at one rotation.



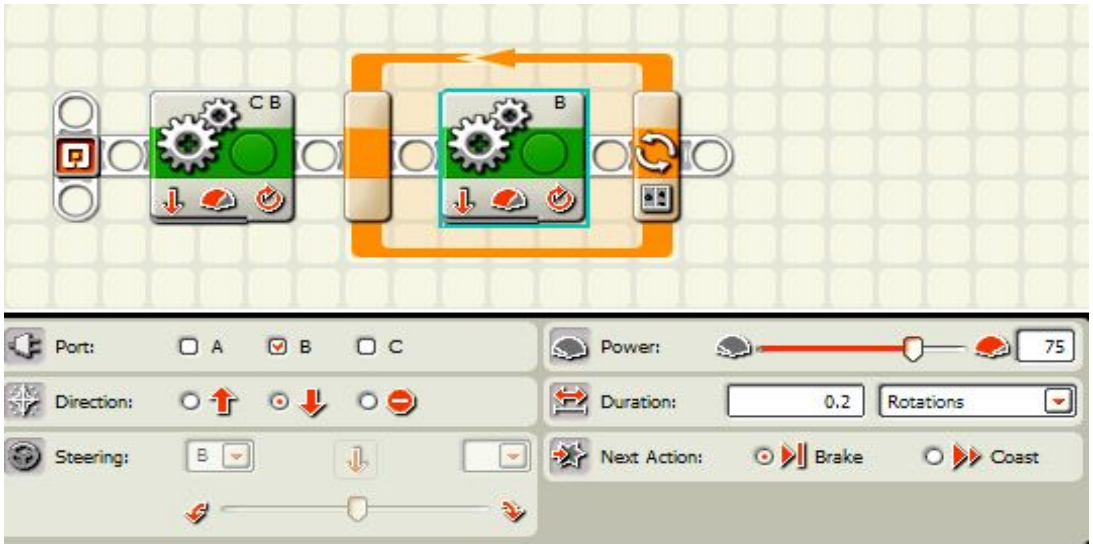
This will pull the robot up against the wall on at least one side.

2. Place a loop on the bar and set it for count and for 3.



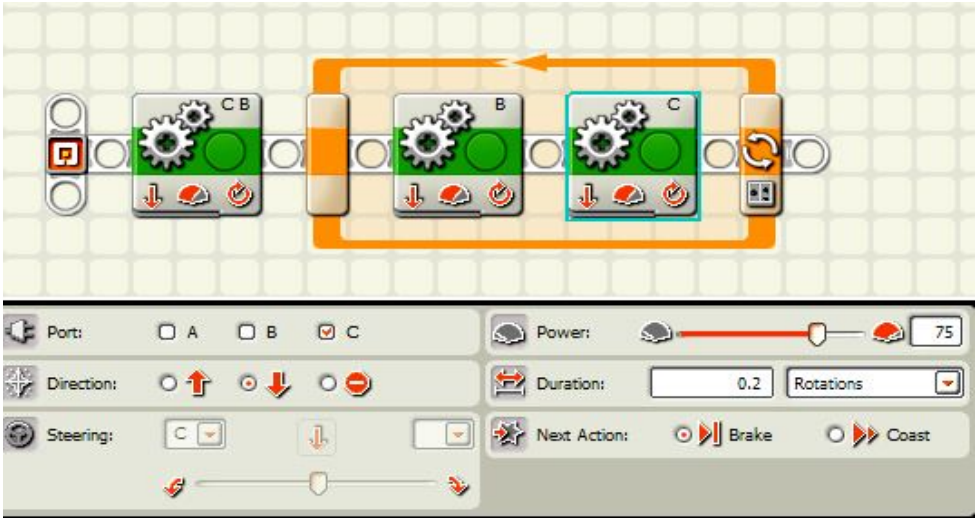
This loop will make the moves you will put in the loop repeat 3 times.

3. Place a move block into the loop, set it to port B and set it for reverse and for 0.2 rotations.



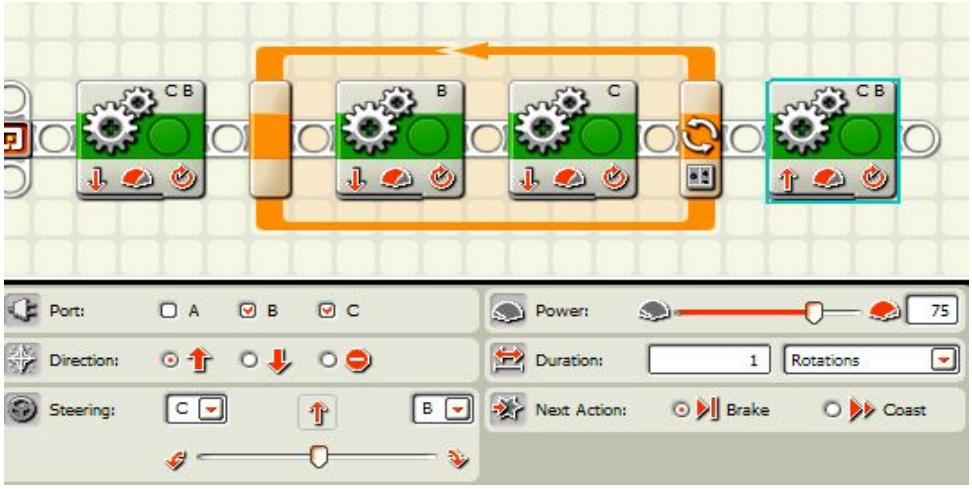
This will cause the robot to turn the motor attached to port B move back just a little bit.

4. Place a move block into the loop, set it to port C and to reverse for 0.2 rotations.



This will cause the robot to turn the motor attached to port C move back just a little bit.

5. Put a move block on the bar and leave it at 1 rotation.



This will make the robot move away from the wall and it should be lined up at a 90 degree angle to the wall.

To review, the robot backs up to the wall not quite at a 90 degree angle but each wheel backs up a little bit to pull both sides of the robot against the wall and it does this three times. It then moves away all lined up at a 90 degree angle to the wall. This helps it get rid of any problems it had with its relation to the walls it may have had.

Front Shimmy

Mission: The robot will back up to a wall going in reverse but not at a 90 degree angle and by alternating between the left and right wheels, it will adjust itself so that it is now pointing at a 90 degree angle to the wall.

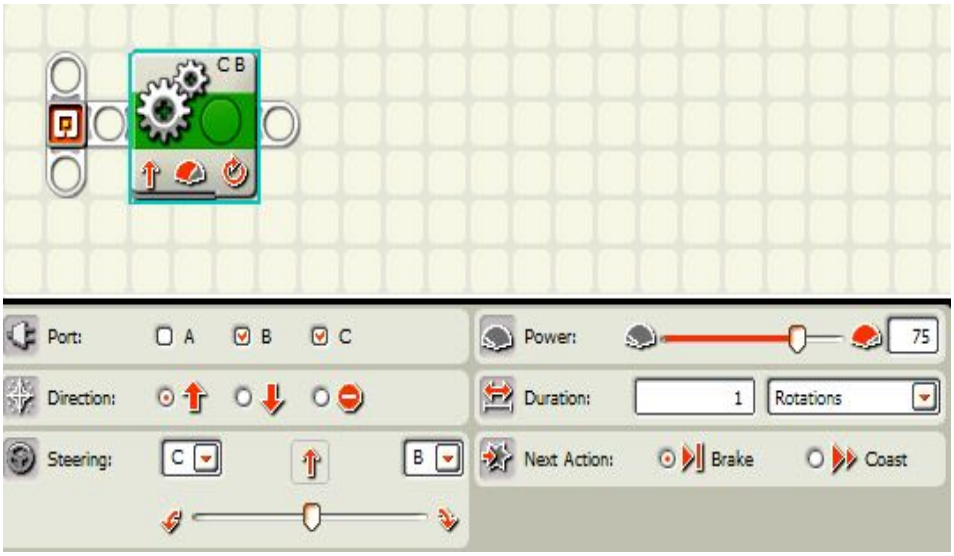
Note: This is a great way to get the robot properly oriented after it has traveled a long distance or made several turns since the robot has the potential to get slightly out of alignment with the walls. If the robot is not getting too out of alignment with the wall, you can have the robot just bump up against the wall.

Equipment:
The robot needs a flat bumper on the front to use to push up against the wall.

Sensors: none

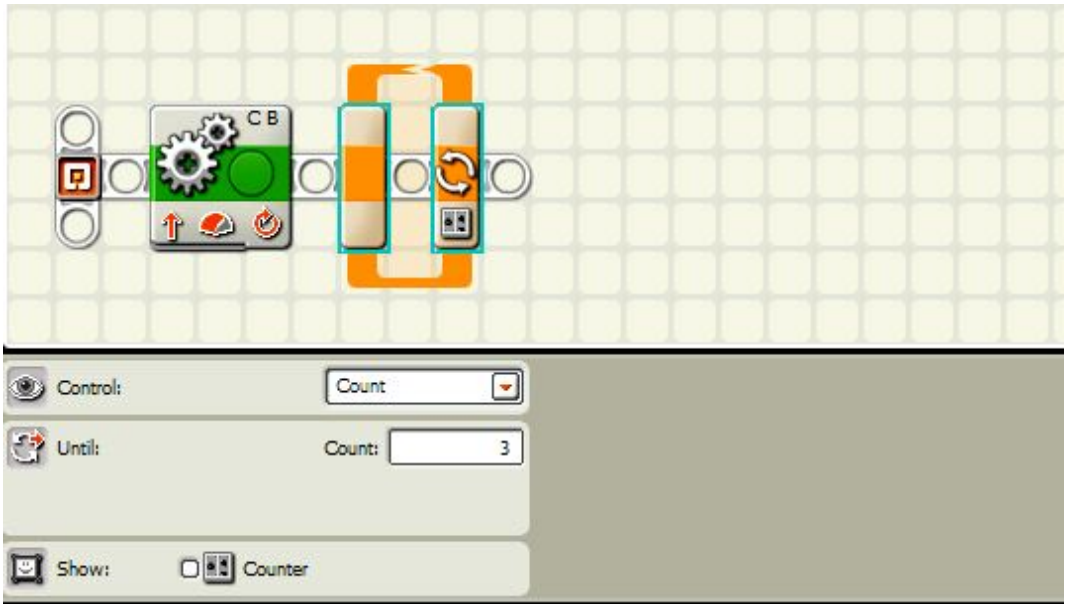
Directions:

1. Place a move block on the program line. Leave it set it for forward for one rotation.



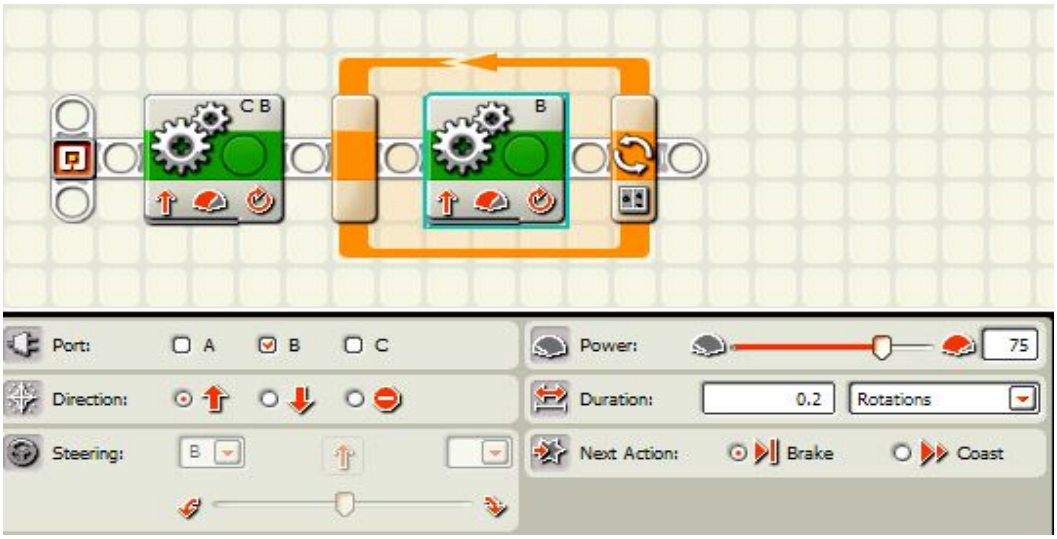
This makes the robot move forward one rotation

2. Place a loop block on the line and set it for count and for 3.



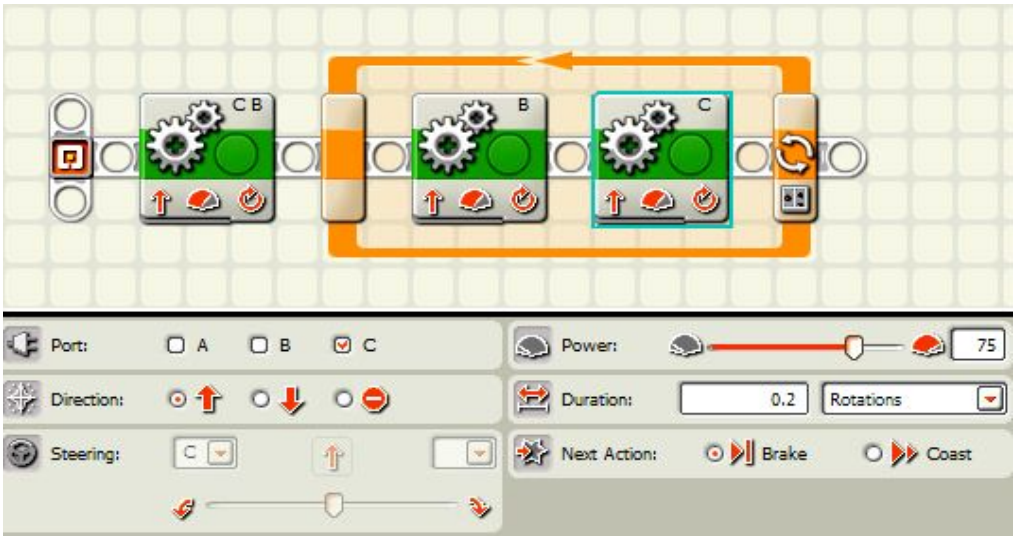
This loop will make the program blocks inside of the loop repeat 3 times.

3. Place a move block in the middle, set it to only port B and set it for 0.2 rotations.



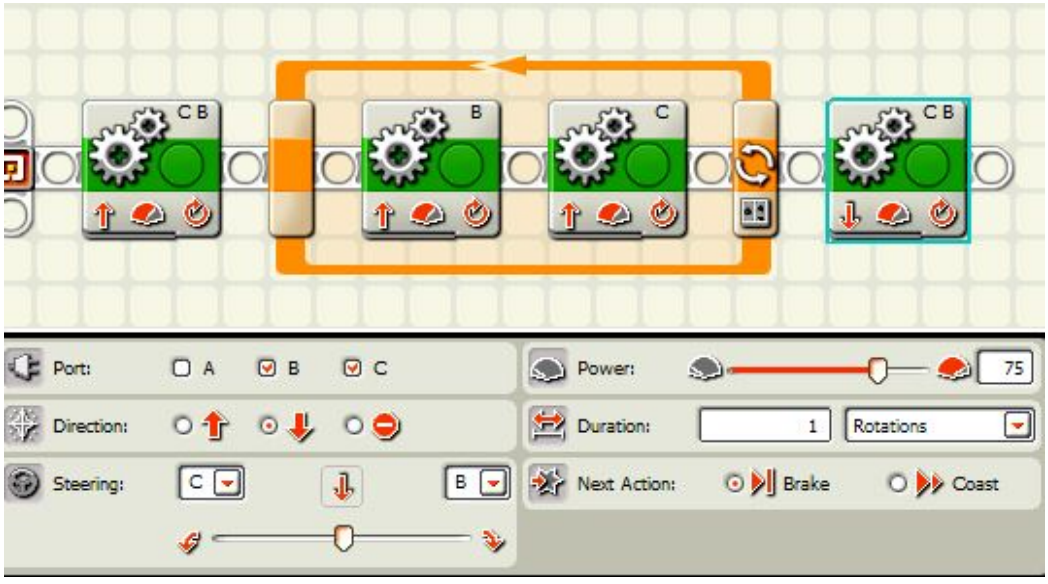
The move block will move the motor attached to port B just a little bit and will help pull the robot closer to the wall on this side if it is not against the wall already.

4. Place a move block in the middle, set it to only port C and set it for 0.2 rotations.



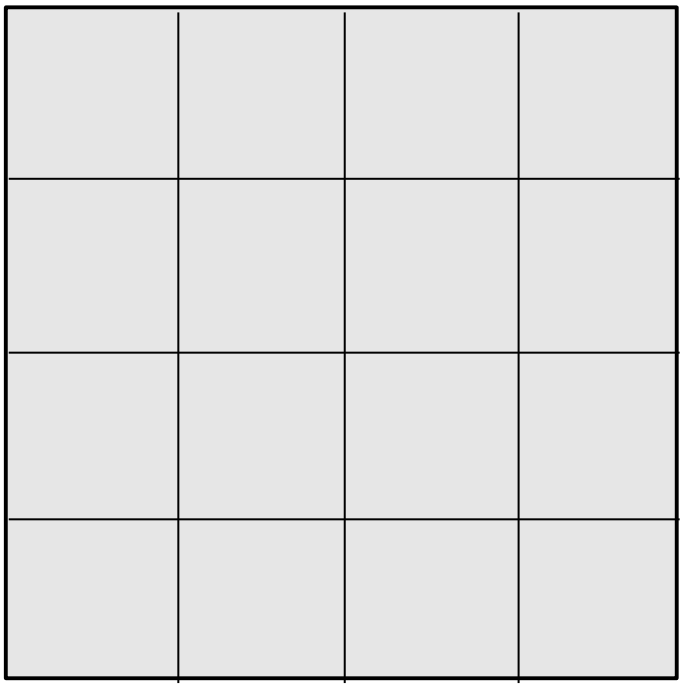
The move block will move the motor attached to port C just a little bit and will help pull the robot closer to the wall on this side if it is not against the wall already.

5. Set a move block on the line and set it for reverse and leave it at one rotation.



The program will run like this: the robot will back up against a wall, then it will run each wheel just a bit three times in a row, and then it will move forward. It should move the robot at a right angle to the wall.

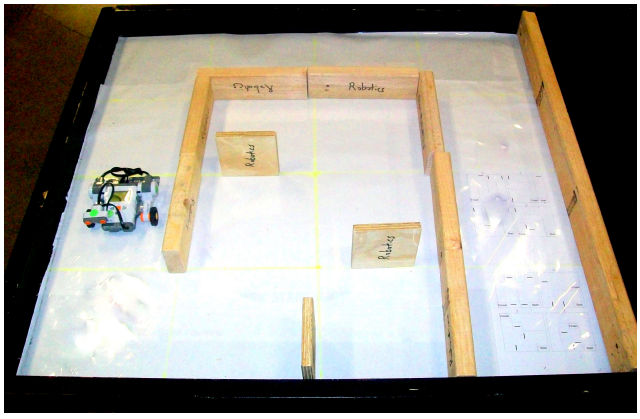
Mazes *An Explanation*



The next few pages show several mazes that can be a real challenge. To keep things easy and low maintenance, there are a few things to do.

Make a paper mat to put under the clear plastic sheet and mark it out with a grid that is about eleven and a half inches wide so that it divides the field into four squares across the top and four squares down the side. Mark the lines with a yellow or highlighter to make them easy to see up not mess up the light sensor if you use the paper for other assignments that use the light sensor. The field should look

Next, you need to make some movable walls for the mazes. These are made from two by fours cut into eleven and a half inch lengths. It isn't too hard to find someone with a saw and you may be able to get the hardware store or the home improvement store to cut them for you. They may charge you to do this but that is the price of convenience. You may just be lucky enough to have some shorter pieces lying around. They don't need to be anything fancy. You will need nine of them.



The knock down blocks are made from one by sixes cut into five and a half inch lengths. You will need at least three of them. It is always good to have some spares.

I labeled everything “for robotics” to keep my middle school woodworkers from using them as sawing a drilling practice and then pleading ignorance when I ask about it.

Put the paper grid under the clear mat so the students can see them and rearrange the mazes as they need to work on their current assignments.

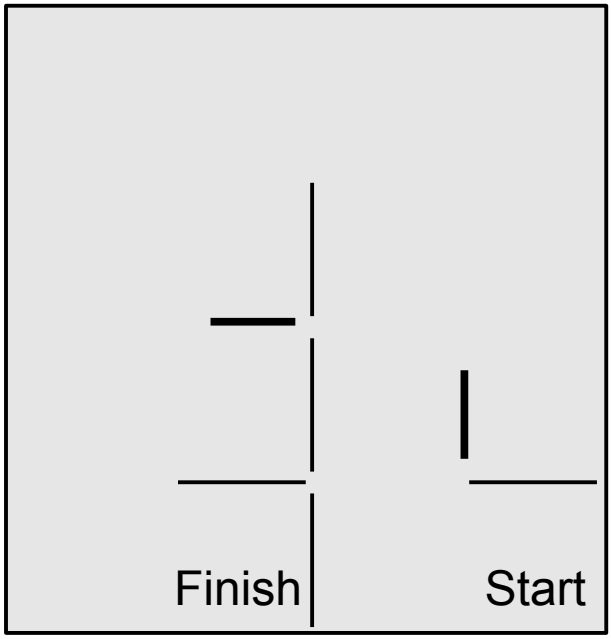
MINDSTORMS *Made Easy*

Maze Rules

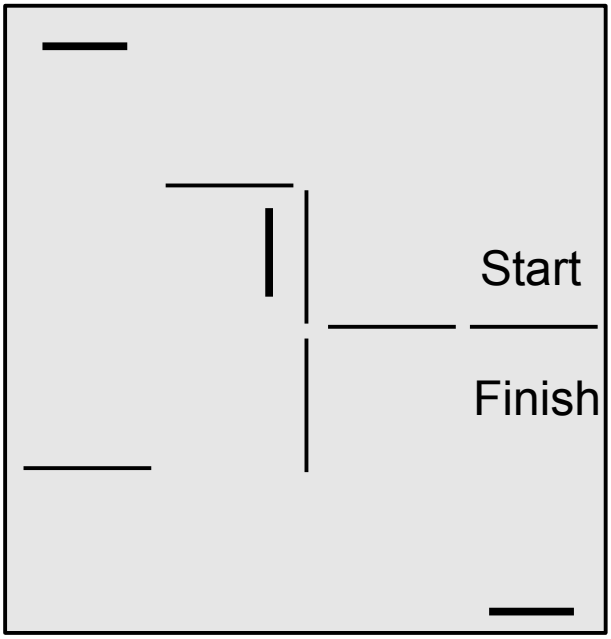
- All wheels must be in the start square when the robot starts.
- No one can touch the robot once it starts.
- No leaning on the table once the maze starts.
- You may hold the wall blocks so the robot can't move them.
- If you do touch the robot, you must start the maze again.
- You may remove a bump block after it is bumped.
- All wheels must be in the finish square at the end of the maze for full credit.

24 Mazes

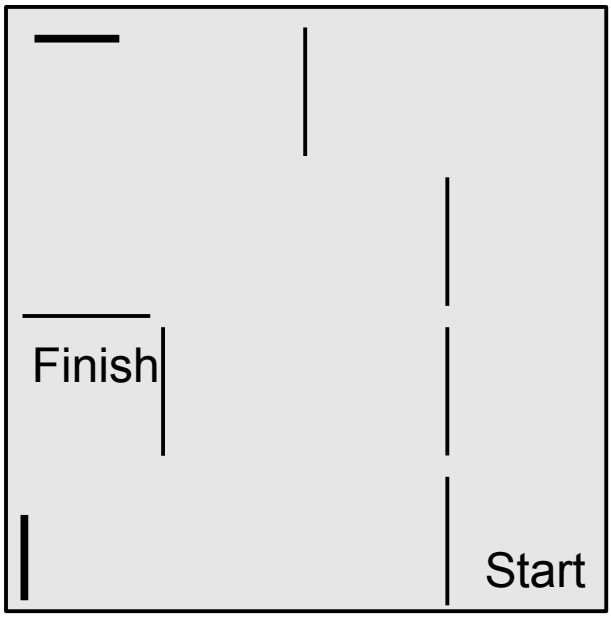
MAZE 1



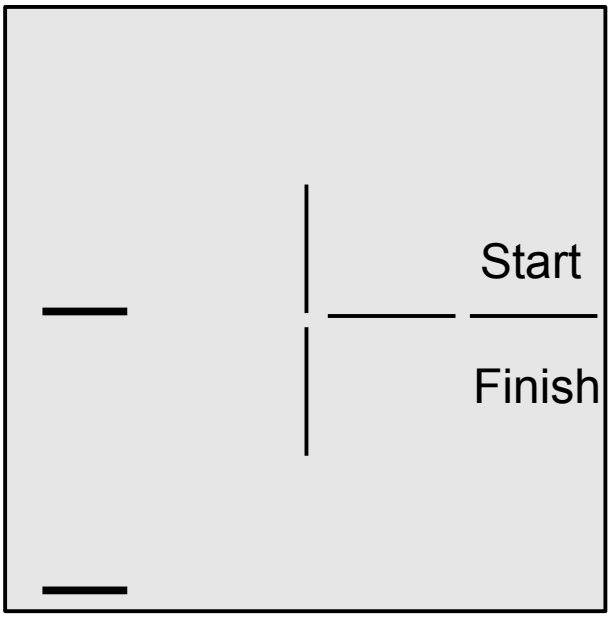
MAZE 2



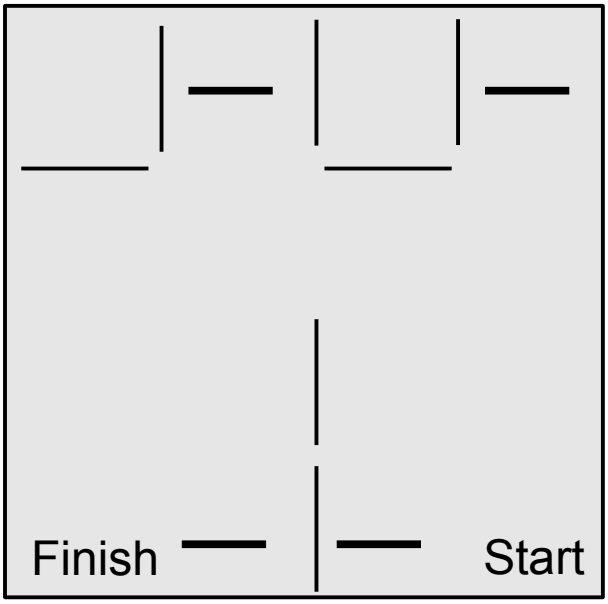
MAZE 3



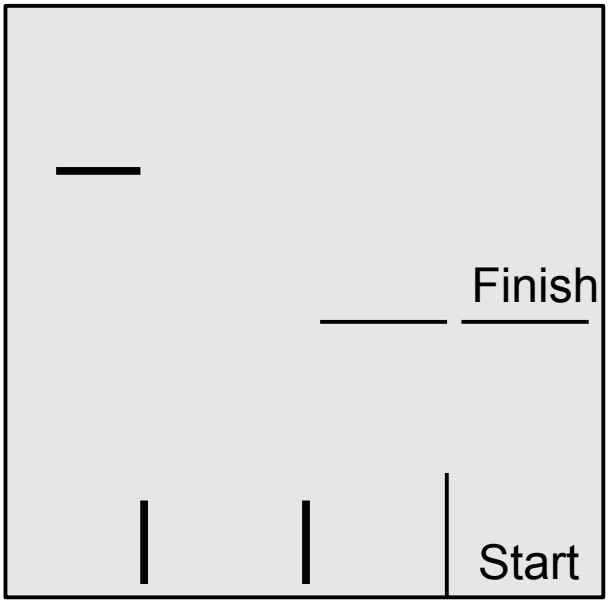
MAZE 4



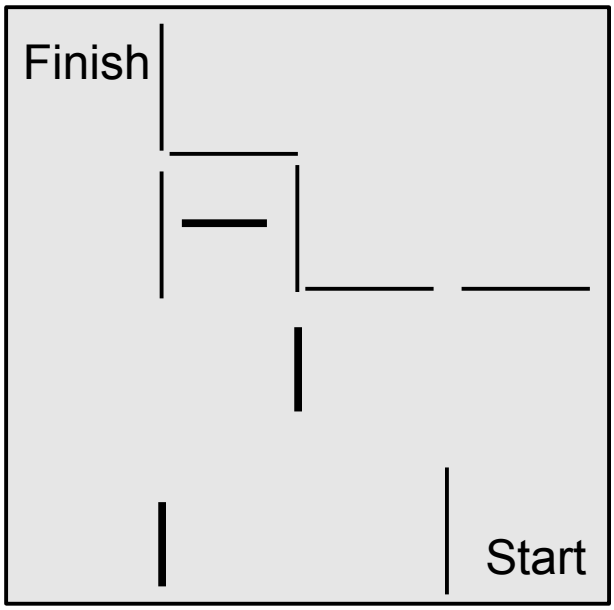
MAZE 5



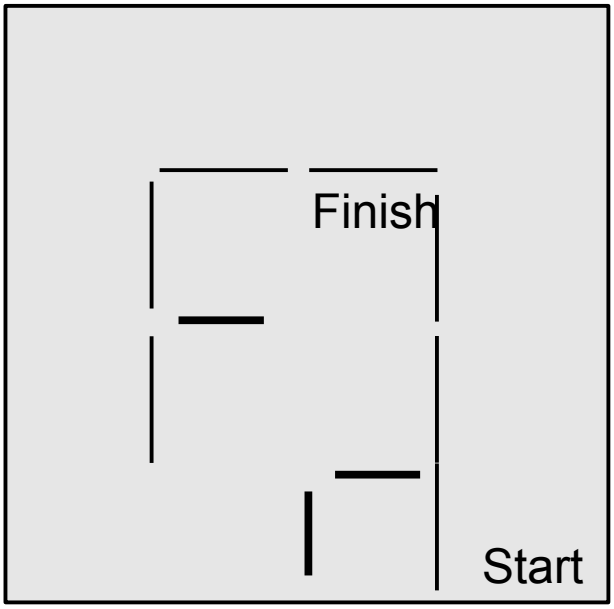
MAZE 6



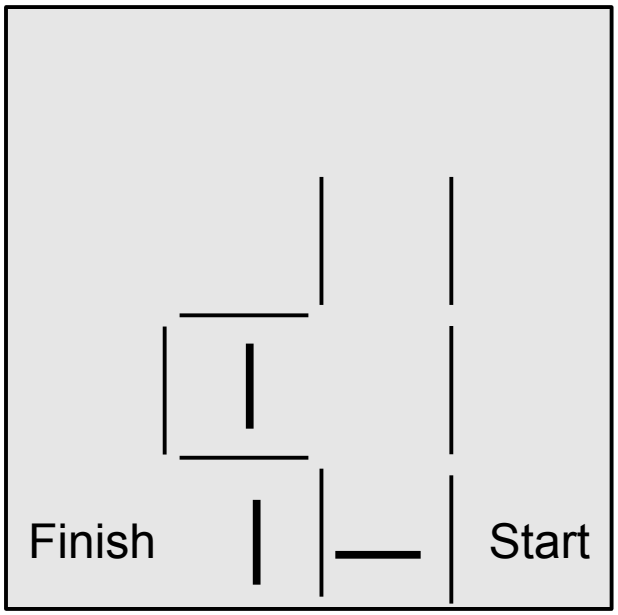
MAZE 7



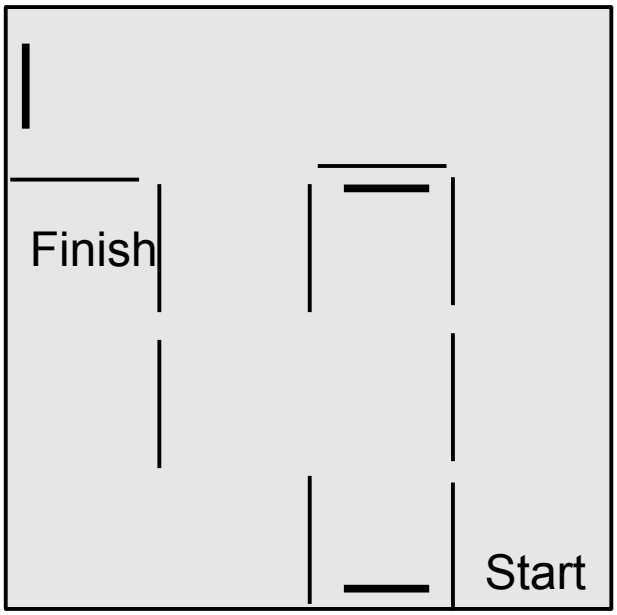
MAZE 8



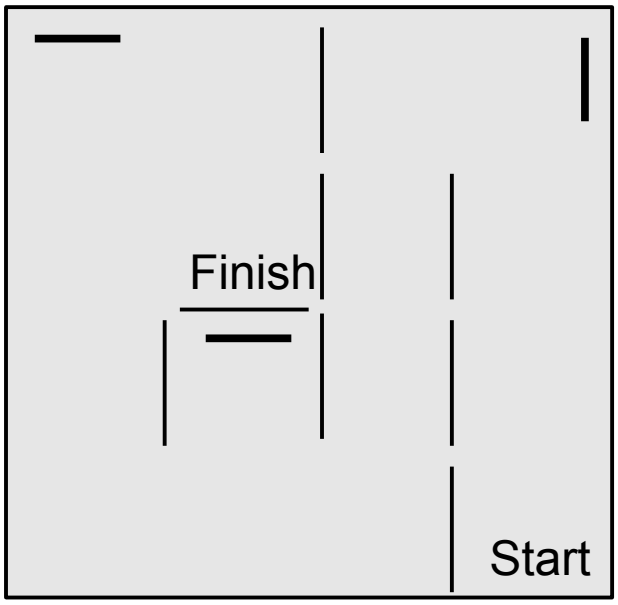
MAZE 9



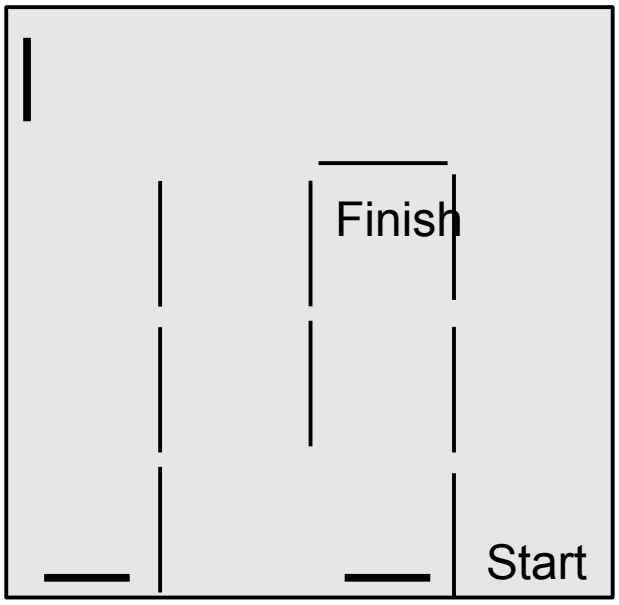
MAZE 10



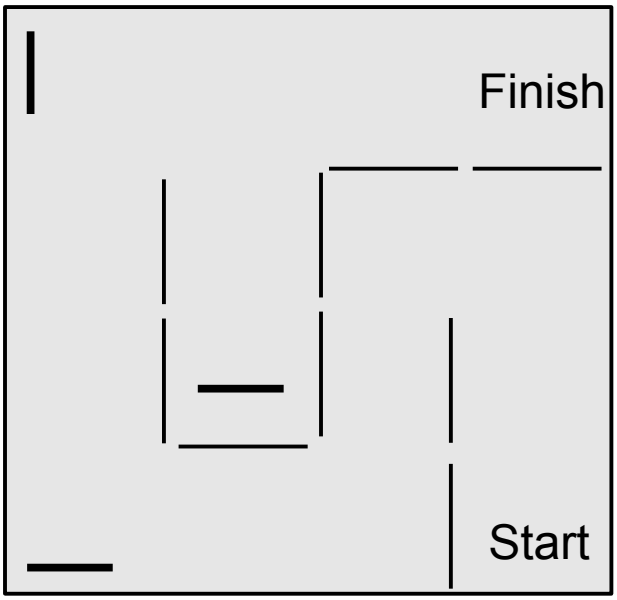
MAZE 11



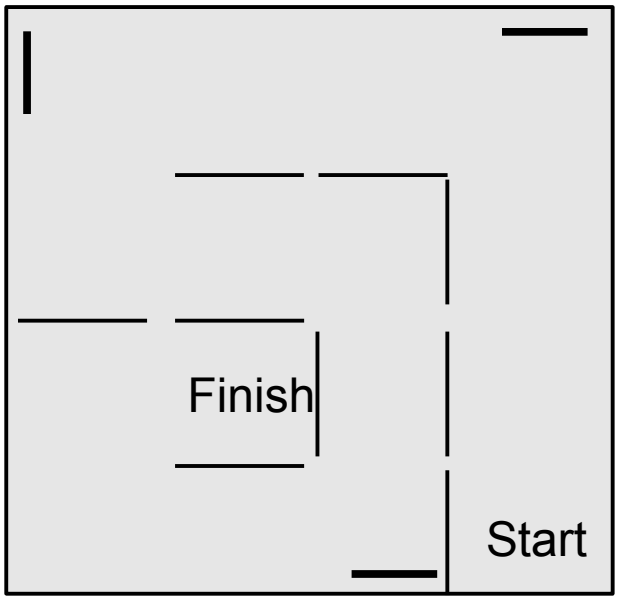
MAZE 12



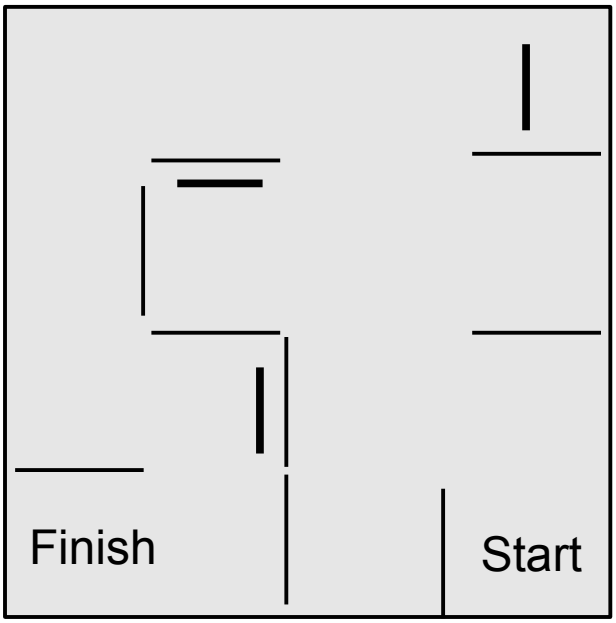
MAZE 13



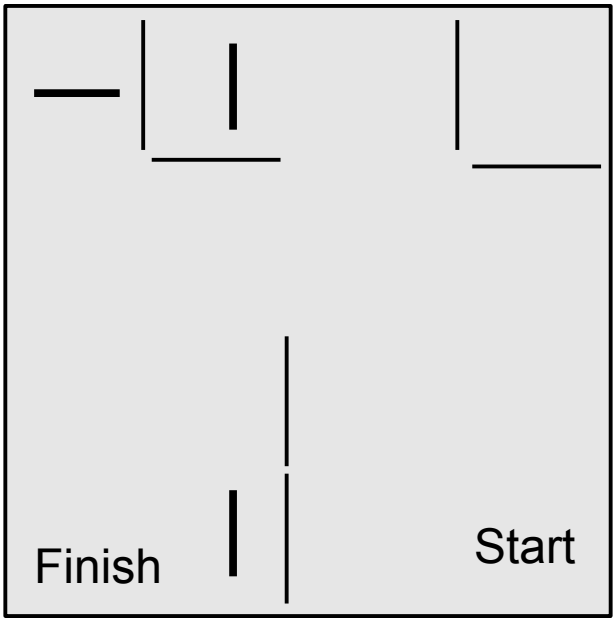
MAZE 14



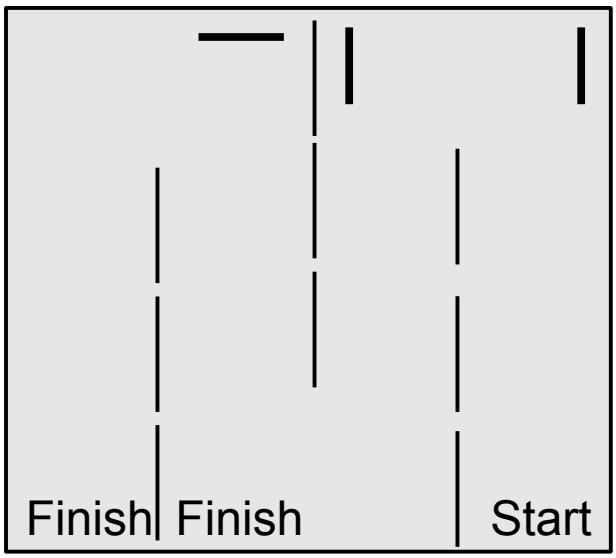
MAZE 15



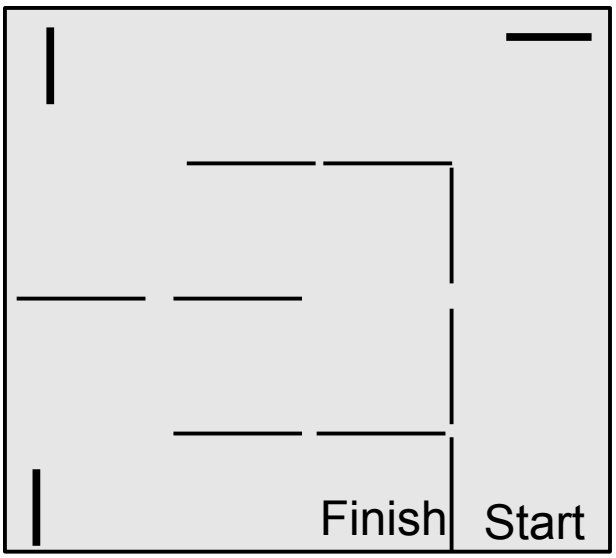
MAZE 16



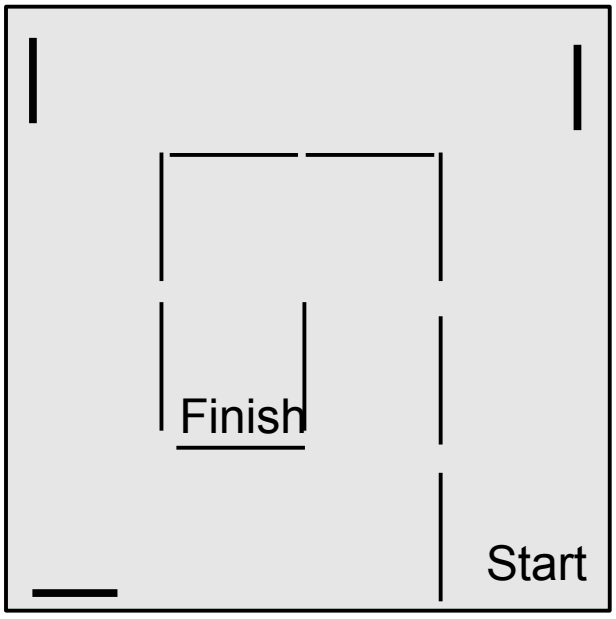
MAZE 17



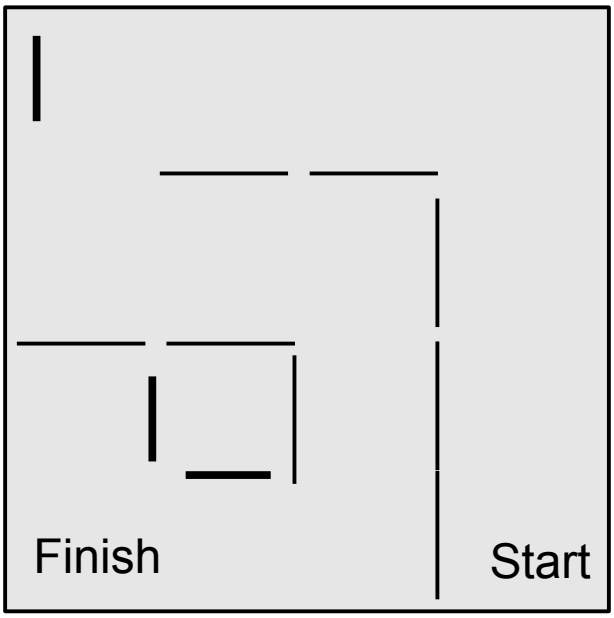
MAZE 18



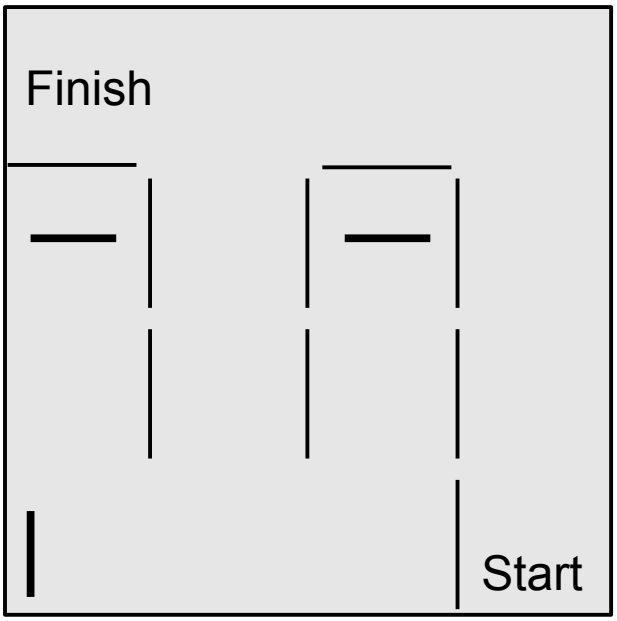
MAZE 19



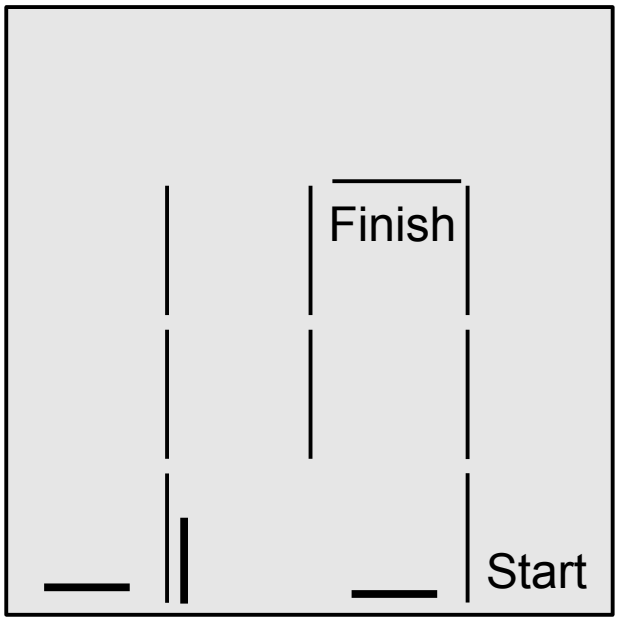
MAZE 20



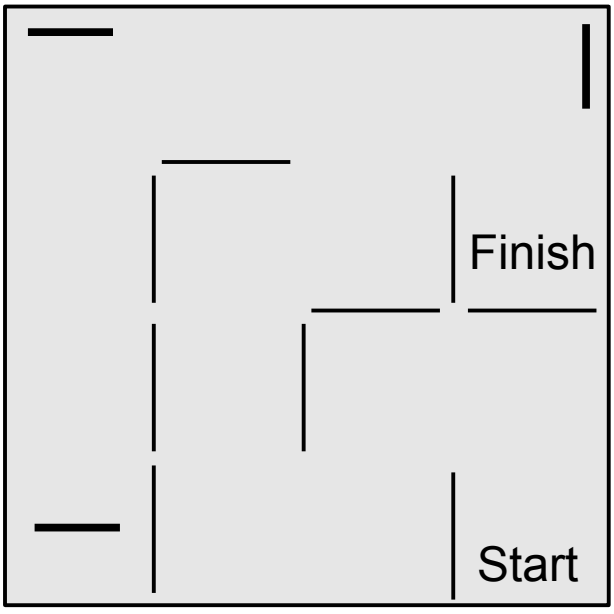
Maze 21



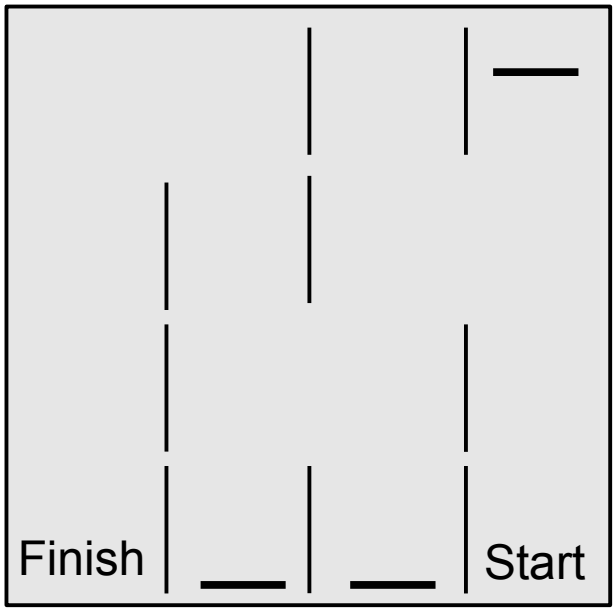
Maze 22



Maze 23



Maze 24



What is a Robot?

What do you think of when you think of a robot? Do you think of walking machines that can talk and have eyes? Do you think of some kind of character from a science fiction movie like C3-PO or R2D2? How is a robot different than a machine that can make decisions? After all, a washing machine can sense water temperature adjust the amounts of hot and cold water to reach a certain temperature, adjust the spin cycle by how the weight is distributed in the tub, but most of us would not think of a washing machine to be a robot because it does not seem like a mechanical animal or person. Many robots used in factories only have a mechanical arm.

Most of the robots we hear about can be divided into two different types: **remote control** and **autonomous (**pronounced "**u-TON-u-mus")**.

Remote control robots are run by a human operator somewhere. It is like a remote control car that you control and it does what you tell it to do. Many of the military's flying robots have human pilots. Many of these robots may be half way around the world while the pilots sit in a chair and pilot the robotic planes from a military base in the U.S. They are basically a very sophisticated remote control airplane. They do not pilot themselves. Autonomous robots use a computer and a program to do things. Right now, you can buy a robot vacuum to clean your carpets. It uses a timer tell it when to start cleaning. It moves around the floor vacuuming up dirt. It uses sensors to know when it comes to a wall, and it turns and goes a different direction. The owner just needs to clean it out once in a while. It is automatic because it runs itself and makes its own decisions.

Robots use sensors to get information about their environment, make decisions about that information, and then use that information to do something. This is sometimes boiled down to this:

Many missions in this book do not use sensors so the robot does what it is programmed to do even if it crashes into a wall or stops short of the finish line. One way for the robot to find its way around is to use a touch sensor so it knows when it comes to a wall. This is how the robot can **sense** what is around it. Using sensors to get information into a robot is called **input**. Any information that goes into a robot is input because it goes *in* to the computer brain of the robot. It can be a light sensor reading the light level on the practice field or the sound sensor reading the level of noise around the robot. It can also be buttons being pushed on the front of the intelligent brick.

Next, the robot uses the data (information) it got from the sensors to **decide** what to do. It can do whatever you program it to do. You may have programmed it to turn when it hit a wall so it turns when the touch sensor gets pushed by moving into the wall. This is where the computer chips inside the intelligent brick does its work. It makes decisions

on what to do from input from the sensors, and makes decisions according to the program, but there is one more step.

Last, the robot uses the program you wrote and any sensory inputs to make decisions and the robot brain then sends **output**. In other words, it sends out a message to do something or **act**. Maybe the robot turns right when it senses a white surface under the robot and turns left when it senses a dark surface under. It might hit a ball when it comes close to it. Most of the time in these lessons the output is a signal to the motors to turn the wheels, but the output can be also a sound to make or a picture or words to put on the display screen. You can see the smiley face on the cover of the book. This is a form of output too.

These three things, **sense, decide,** and **act,** are shown by other things a robot might be programed to do. For instance, it may move forward on a table until it senses the edge of it, then back up and turn, and continue forward until it senses the edge again then back up and turn and keep repeating the same actions over and over again. The robot does something (move forward) until it senses something (the edge of the table), it makes a decision to act (back up and turn) and then does it.
The term “robot” means many things to many people. Sometimes a remote controlled vehicle or plane can be called a robot, but a little bit more technical definition of a robot is possible. A robot needs to have four elements. One, it needs to be a **machine**. Two, it needs to be automatic. This means that it needs to be able to work without someone controlling it. This is why a remote control airplane or car is not technically a robot. Three, it needs to be **reprogrammable**. In other words, it needs to be able to do more than one job. To use the example of a dishwasher again to see how well it fits these three rules. It is a machine, it is automatic (you don't need to push a button to tell it when to stop washing and start rinsing), but it is not reprogrammable. It will only wash dishes. You cannot program it to paint a car or put together a puzzle, so dishwasher is not a robot. The fourth element is that it needs to be **responsive**. That means that the robot must use sensors to relate to its environment so it knows if it bumps into a wall by using a touch sensor or if it is close to an object using an ultrasonic sensor, or if it is following a dark line using a light sensor. Some robots use GPS signals to guide them on paths that are miles long. These are examples of robots being responsive. They have ways of keeping themselves on track and can sense if they are touching something or even what color something is.

The term a**ndroid** refers to robots that have a human likeness. They have all the characteristics of a robot—machine, automatic, and reprogrammable—but also have the added step of looking like a person. 3C-PO from the Star Wars movies is a good example. He does not look exactly like a person but his basic shape is the same.

What is a Robot? Questions

Name_____

Period_____

Define these terms:

- input
-
- output
-
- sensor
-
- automatic machine
-
- robot
-
- android

Answer these questions:

What four characteristics does a robot have to have?

1.
2.
3.
4.

Is a dishwasher a robot? Why?

Is a garage door opener a robot? Why?

Is a Roomba vacuum cleaner a robot? Why?

Is a Roomba vacuum cleaner an android? Why?

The History of Robotics

Stories of mechanical devises made to look like people or animals have been around a long time. In the Iliad, there is a story of the god Hephaestus making mechanical hand maidens out of gold. A story from China tells of an man presenting the king a life-sized, human-shaped figure filled with his mechanical handiwork. In medieval Europe, a Dutch tale describes mechanical birds and angels making sounds by using a system of pipes.

These machines were made to run with gears and springs like old fashioned clockworks or wind up toys. The mechanical devices that performed preset actions by following a set of instructions. A device like this is not a true robot. It is called an automaton (pronounced “u-TOM-u-ton”).

During the 1700's great advances were made in automating the weaving of cloth to create complex designs without the use of skilled labor. These weaving machines were not robots, but the knowledge gained in building the automated looms helped build automated machines in other fields.

The first fully programmable, digital computer was build my Konrad Zuse in1941 in Germany. The first American programmable computer was built in 1944 by Howard Aiken and Grace Hopper. It was called the Mark I and was used by the U.S. Navy. ENIAC was build in1946 and weighed 60,000 lbs.

The mechanical devices or automate (more than one automaton) did not have the ability to change their programming, though. That would have to wait until the invention and development of computers. Computers and sensors allow a robot to be reprogrammed and react to its environment.

The word “robot” comes from a play written in 1920 called R.U.R. orr *Rossum's Universal Robots by the Czech playwright Karel Capek.* He gives credit for his brother thinking up the word. People build robots to help them, but the robots decide instead to kill all the people. The word robot means servant or slave. The robots in the play were not mechanical, though. They were made of flesh and bone like Frankenstein.

Twenty-one years later in 1941, science fiction writer Isaac Asimov first used the word “robotics.” It referred to the technology of robots. He wrote a series of short stories and novels about the rise of robots—some good and some bad—and he also invented the three laws of robotics and later added the “zeroith” law when he saw a need for even a higher law than the first one.

Asimov's Laws of Robotics are:

0. A robot may not harm humanity, or, by inaction, allow humanity to come to harm.
1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence, as long as such protection does not conflict with the First or Second Law.

In 1956 George Devol and Joseph Engelberger (also known as the “father of robotics”) started the first robot company. They started to manufacture industrial robots to use in factories. The first robot ever used in a factory was was in 1961 and it was used by General Motors in an automobile factory. Today, many factories use robots for welding, moving materials, and painting. These robots do not look like people or animals. They look something like an arm.

Disneyland started in the 1960's using various automata to use instead of actors in some of their rides like on Meet Mr. Lincoln, The Pirates of the Caribbean, and Splash Mountain. These machines look like people or animals, but they do not react to their environment like true robots do. They just repeat the same few actions over and over; therefore, they are automata.

The first microprocessor was created by Ted Hoff in 1971 at Intel and measured an1/8th inch by a 1/16th of inch Even though it was a fraction of the size of a postage stamp, it was more powerful than the 60,000 lb. ENIAC build 25 years earlier. The advances in computing power has help robots to be more and more complex and inexpensive.

A company named iRobot now makes many different robots to use around the house as well as for the military. They make robots named Roomba that clean your floors. They make robots that can clean out your gutters. They make robots that clean your pool, travel the oceans recording data, and patrolling harbors. They make robots that can search for mines, surveillance robots to examine dangerous situations, and even robots that can pull wounded a wounded individual to safety.

As a matter of fact, robots are often used in places where the work is “dull, dirty, or dangerous.” Robots are good at doing the same thing over and over without needing a break or calling in sick. They can work twenty-four hours a day and seven days a week. They are used in dirty work that people would not like to do. Also, they are very useful for dangerous work. They are used to explore the depths of the sea and vast distances of our solar system. It is very difficult for people to go to the bottom of the ocean or to distant planets. If a robot is damaged and fails to work on a distant planet, it can be a sad situation, but if astronauts die on a distant planet, it would be a national disaster and devastating to all the people involved.

Today, robots are changing the way we do things in all aspects of our lives. They are in factories, businesses, and even in our homes. They are being taught to drive cars, pilot planes, and sail the oceans. They can do many of the jobs people did just a few years ago.

We are in a change of life every bit as big as the industrial revolution when factories replaced farming as the main job for people. Horses and buggies were replaced by the automobile and hand crafted items produced by skilled craftsman was largely replaced by mass produced items built by unskilled labor. It was a big change for society and we are going through something very similar now. Back in the industrial revolution, machines replaced many workers. This was a very difficult time for many people, but the machines also created more jobs. We may complain about the smell of diesel buses and trucks, but very few people would want to go back to horse drawn buggies and wagons and the mess and smell that so many horses would cause in our cities.

We can watch dancing robots on the internet. Robots are driving cars and flying airplanes. Robots are even used to work with autistic children. Robots will be in more places than we ever imagined just a few years ago.

As robots come into the home and in service industries like hospitals, people worry about the possibility of robots replacing human friends and relatives. Who knows what the future will hold? Will it be a smooth change we all will enjoy, or will it be a difficult change that will disrupt our lives? No one knows what the future holds. We will find out when we get there.

History of Robotics Questions

Name_____
Period_____

1. Tell about two of the mechanical devices built before or during the 1700's that looked like animals or people.

2. Define automaton.

3. Who invented the term robot?

4. Who invented the term robotics?

5. What are the four laws of robotics?

0.

1.

2.

3.

6. Advancements in what field have allowed robots to be more programmable, more complex, and more inexpensive?

7. Jobs that are best suited for robots usually fall into three categories. What are the three categories?

1.

2.

3.

8. Robots will replace some workers, but it will also create other jobs. What might some of those jobs be?

Class Rules for Robotics Class

Name_____Hour _____
The fee for this class is _____ . Please take care of this within the first 2 weeks of class by paying at the front office and presenting the receipt to the teacher.

Rules:

1. Students will have all appropriate materials and supplies and will be seated quietly when the bell rings. Students who are not in their assigned seats and/or did not bring the needed equipment will be marked tardy.
2. Respect the people, equipment, supplies, and furnishings. Do not touch, borrow or use other students' items without their permission. No students in the office or storerooms unless otherwise instructed.
3. Cleanup lab and work areas when cleanup bell rings.
4. Talking is to be kept low at all times because the teacher's voice should be heard at normal tone.
5. Follow directions the first time they are given.
6. Observe all rules in the student handbook.
7. Absolutely no horseplay at any time.
8. No food or drink in the classroom or workshop. If it is brought in, it needs to be thrown away.
9. No electronic devices
10. Presenting other people's projects as your own for a grade is cheating and will be punished.
11. Destruction of tools and/or supplies is vandalism and can be punished with suspension and possibly expulsion from school.
12. Theft of tools and/or supplies is illegal and can be punished with suspension or expulsion from school.

Discipline:
Discipline will follow the rules as outlined in the student handbook. Typical actions include warnings, writing assignments, written apologies, calls home, referrals to the office, in school suspension and removal from class.

Students: I have read this list of rules, understand them, and agree to follow them.

Student Signature_____

Date_____

Parent or Guardian's Signature_____

Date_____

Bell Ringers

Many classes start with a writing prompt to give the teacher a chance to take the role and to give the students something quiet to do before the bell rings so they will be ready to learn. Here is a list of writing prompts you can use to do good bell ringers.

1. What do you hope to learn about robotics in this class?
2. Explain how to start a new program in NXT-G.
3. Explain how to use the Move block.
4. Explain how to use a Move block to make the robot go in reverse.
5. Explain how to use the Move block to make a spin turn.
6. Explain how to make a wheel stop turning.
7. Explain the difference between a stop and a coast in a Move block.
8. Explain how to write comments above the blocks in the program.
9. Explain how to make a pivot turn.
10. Explain what the rotation sensor does that is in the motors does.
11. Explain how a rotation sensor helps with running a robot.
12. Explain what a flowchart is and how it helps programming.
13. Explain what ports does the Move block default to in order to attach the motors.
14. Explain how to save your programs.
15. Explain what pseudo code is and how it helps to write a computer program.
16. Explain how the ultrasonic sensor works. How does it know how far away an object is?
17. Explain how the ultrasonic sensor works better on flat objects than on round or irregular surfaces.
18. Explain to which port the touch sensor defaults.
19. Explain to which port the ultrasonic sensor defaults.

20. Explain to which port the light sensor defaults.

21. Explain how the ultrasonic sensor is better in some situations than the touch sensor.

22. Explain how a Loop block works and how it saves time in programming.

23. Explain how a Switch block works and how it helps in programming.

24. Explain the various types of Loops there are in NXT-G.

25. Explain the various types of Switches there are in NXT-G.

26. Explain how to set the level of a light sensor so that it can sense a dark line.

27. Explain how you would figure out how many revolutions it takes to go 3 feet.

28. Explain how setting a move block to unlimited and using a touch wait block after it can be useful in programming.

29. Explain how setting the robot to go too quickly can be a problem when using a light sensor to make the robot stop on a dark line or follow a dark line.

30. Explain how to use a wall to straighten out your robot when it is running a maze.

31. Explain why it is better to bump into a block to knock it over and then backing up is better than running over a block.

32. Explain what you can add to your robot so that it won't catch a wheel on the walls and make the robot turn when you don't want it have it turn.

33. Explain which sensor you like to use better—the touch sensor or the ultrasound sensor.

34. Explain one thing you wish robots could do to make your life easier.

35. Explain how robots have made it cheaper to manufacture things.

36. Explain what “autonomous” means.

MINDSTORMS

Made Easy

NXT-G Programming

Certificate of Graduation

This certificate acknowledges that

_____

has completed the Mindstorms Made Easy Curriculum and has mastered the concepts in its course of study.

The day of ___ of _____ in the year _____.