Building a Robot

There are several components to a mobile robot (one that moves around).

1. An NXT brick.

2. Fully charged battery pack or 6 AA batteries.

3. Motors (1 or 2 to drive the robot)

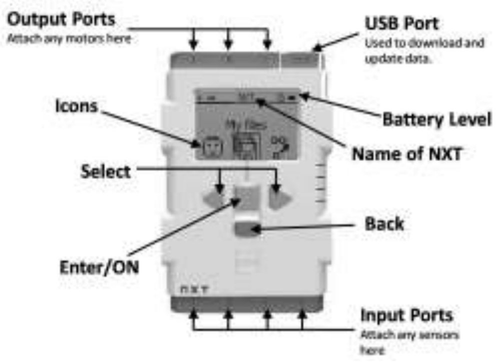
4. USB connector cable (if you have an NXT without Bluetooth capabilities, as shown here)

5. Cables to connect the NXT to the motors or sensors

6. Wheels or treads

7. Axles, bricks, connectors, plates, and other LEGO® parts.

Let's review the different parts of the NXT



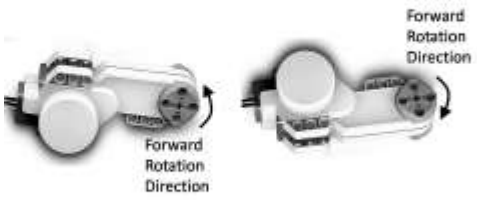
1. The cables connect to ports A, B and C for **motors** (outputs) and ports 1, 2, 3 and 4 for **sensors** (inputs).
2. To download and update programs, ensure the NXT is on and connect the USB cable between your NXT and computer's USB port.
3. Be sure to keep the NXT viewing window and buttons free when creating a robot. This will allow you to access the files on your NXT without having to dismantle your creation. Also be sure to keep the ports free to allow for easy sensor and motor connecting
4. Also note that there is a charge port on the end of the NXT, just below the sensor ports (if your NXT window is facing up). It would be wise to keep that hole free so that you may recharge the NXT without having to remove it from your robot each time you need to charge.
5. If you do not have a rechargeable battery pack for your NXT, you should make a design that allows for an easy removal of the NXT to change the AA batteries when necessary.

Motors and Wheels:

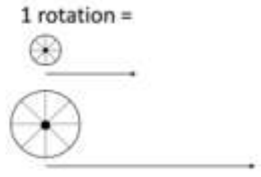
The simplest robots are driven by two motors, each attached to a wheel on opposite sides of the robot. This simple design is both easy to build and control with the NXT.

Things to Remember with the Motors:

1. The "forward" and "reverse" directions of the motors are pre-determined by the NXT:



   You will have to adjust the design of the robot so that what the NXT deems as a "forward" rotation also moves your robot forward.
2. If you have already maxed out the speed setting on your NXT, you can get your robot to move even faster by trading your wheels for larger ones. If the circumference of the new wheel is larger, then the distance traveled per motor revolution is also larger.



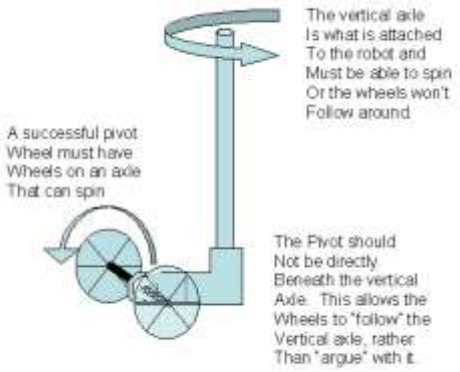
3. It is **VERY IMPORTANT** to provide proper support for your NXT brick and motors.

THE MORE KNOBS OR PINS CONNECTED BETWEEN TWO PIECES, THE STRONGER THE CONNECTION!!!

For additional tips on building a stronger robot, please see our **building tips** page.

Making a robot that can move around:

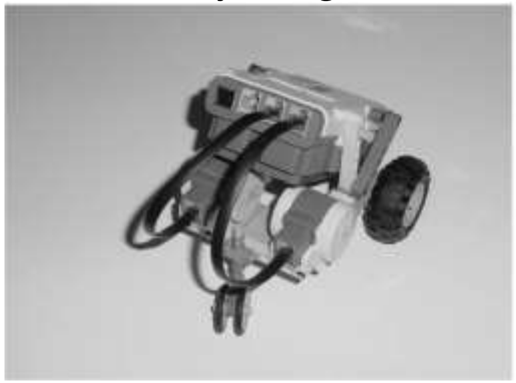
Most robots have 3 or 4 wheels. If you don't care if it can move right or left, then simply attach one or two free spinning wheels on the front or back of your robot. However, if you want to have a robot that can turn you must have a **PIVOT WHEEL**. Think of the front wheel on a shopping cart; if it doesn't turn then you can't easily turn the shopping cart. Pivot wheels should always be in the "back" of your robot.


The vertical axle Is what is attached To the robot and Must be able to spin Or the wheels won't Follow around
A successful pivot Wheel must have Wheels on an axle That can spin
The Pivot should Not be directly Beneath the vertical Axle. This allows the Wheels to "follow" the Vertical axle, rather Than "argue" with it

There are any numbers of ways of attaching the vertical axle to the robot, but you have to have a hole that it can run through. Using connectors, plates with holes, or any other pieces create a hole and secure the axle through the hole using bushings (round pieces that slide on and lock onto the axles).

NOW MAKE A ROBOT that has

- One RCX
- Two motors
- Two wires
- Two wheels attached to the motors
- A pivot wheel on the front (or back)
- Make it sturdy enough not to fall apart.



This robot has two motors mounted below the NXT that each drive separate front wheels. Extra studless beams and pins were added to create a more sturdy chassis, and a pivot wheel is located in the back to allow the robot to turn easily.

Building Tips

Pins:

There are a couple types of pins you should be familiar with.

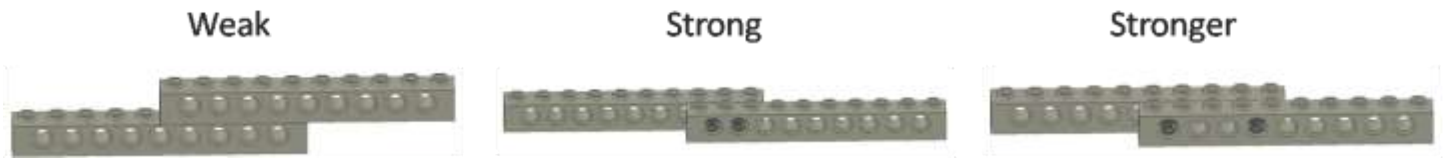
- Friction Pins: these pins are **black** and come in both two and three-hole widths. These pins add resistance to any spinning motions to the connecting pieces to spin with respect to each other.
- Non-Friction Pins: these pins are **gray** and allow the connecting pieces to spin freely with respect to each other.
- Axle Pins: Use these pins when you need to connect a piece with a round hole into a piece with an axle hole, such as a beam to a gear.

Connecting Beams:

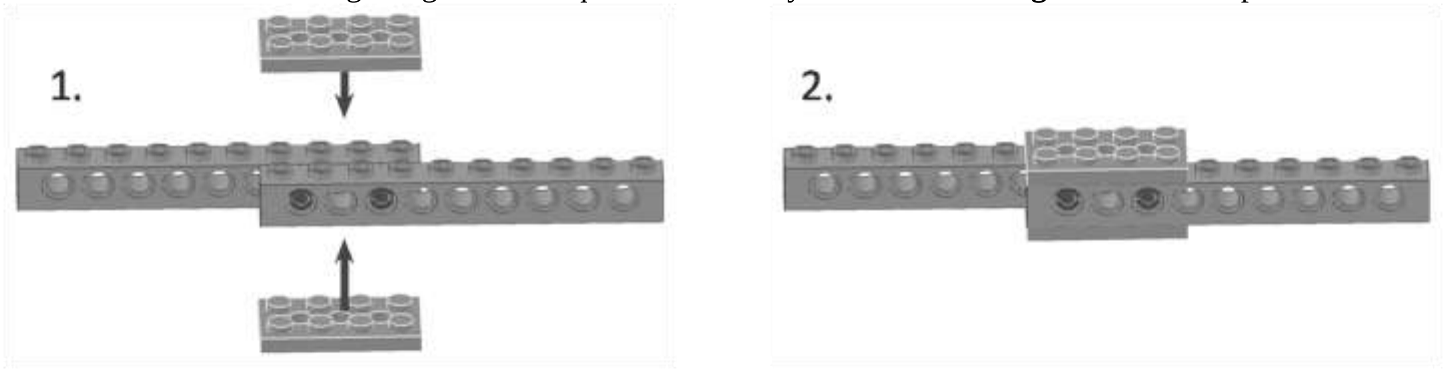
When connecting a pair of beams together, there are techniques you can use to make a connection that is nearly impossible to break apart. If your beams are strongly connected, you will have less unwanted bending and movement.

If you looking to extend the reach of your Lego robot, there are a couple of techniques you can use:

1. Using only knobs to connect the beams is a very **weak** connection and should be avoided whenever possible
2. Connecting the beams using only pins is **stronger**. Obviously, using only one pin allows the beams to spin independently of each other. If this is what you want, great! If you are looking to stabilize a structure, however, it is best to use <u>at least</u> two friction pins (more is better). The further apart you place the pins, the stronger your beams will be connected. Using only pins will prevent twisting from happening between the beams, but will not prevent the beams from being pulled apart.

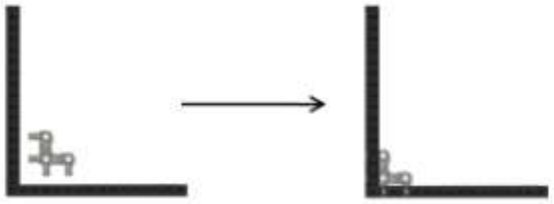


3. Sandwiching is a good technique to use when you want the **strongest** connection possible



Making Corners:

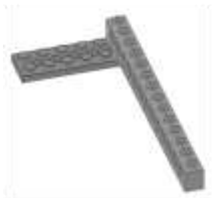
The easiest way to make a strong corner is by using a special corner piece:



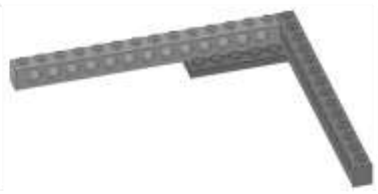
If you do not have access to such a corner piece, here is an alternative way to create a strong corner:

1. Take one beam and attach it along the width of the plate (two knobs wide **minimum**)
2. Place another beam lengthwise on top of the plate to create a corner
3. Connect an additional brick on the inside of the corner, on top of the plate.
4. Place a final plate on top

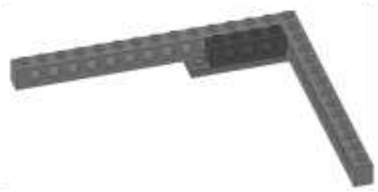
1.



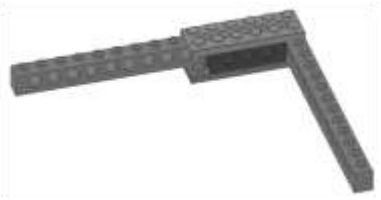
2.



3.



4.

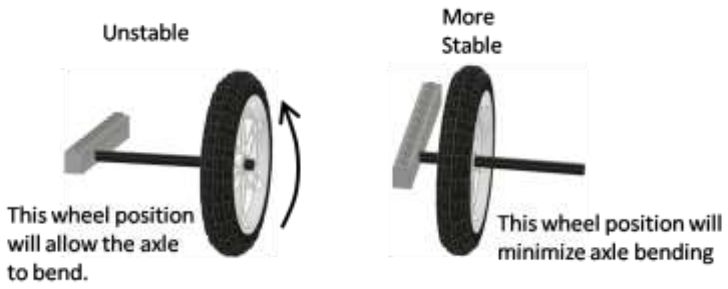


REMEMBER: connecting more knobs, using additional pins, and sandwiching will lead to a very sturdy robot.

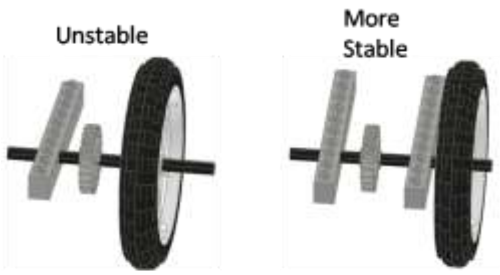
Connecting Wheels:

If you are creating something with wheels, using certain techniques can greatly increase the stability of your robot.

1. Wheels should be attached as close to the nearest beam as possible. This prevents the axle holding the wheels from bending when weight is added.


Unstable
This wheel position will allow the axle to bend.
More Stable
This wheel position will minimize axle bending

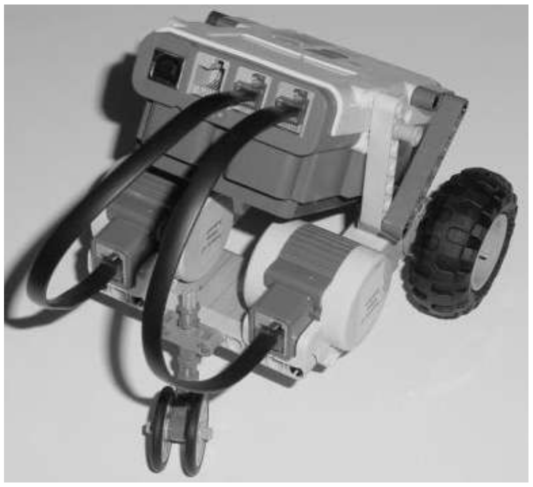
2. If you are using gears as well, be sure to place supports on both sides of the gear before attaching the wheel. Without supports on both sides, you can run into problems with gear to gear connections.



Pivot Wheel

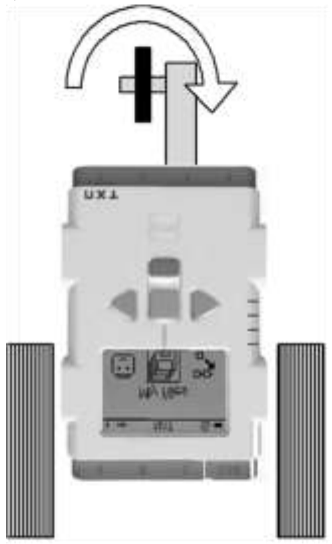
Building a Sturdy Robot:

When you make a robot with wheels (rather than with treads) you will likely steer the robot using the two drive wheels and motors, but in order for the robot to actually turn, the front wheels must be able to turn as well. The picture below shows a robot with two motors that drive the rear wheels and a small pivot wheel under the front of the robot.



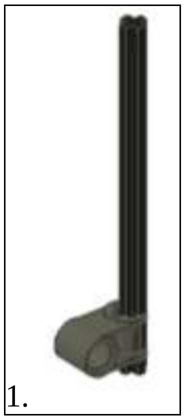
What does the pivot wheel do?

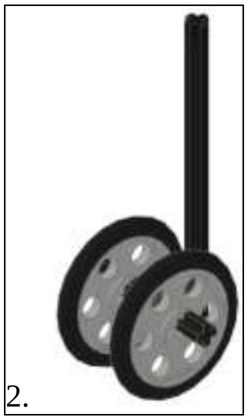
1. The pivot wheel will spin around whenever the robot turns.
2. The wheel is a "following" wheel and will follow in the direction the robot is moving.

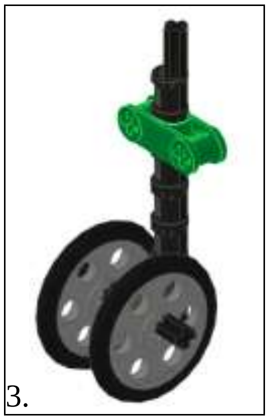


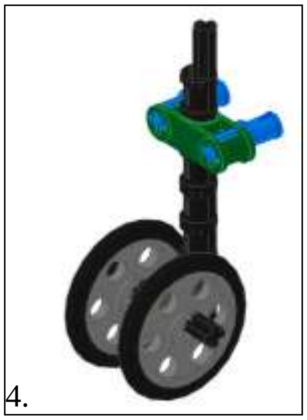
Building the Pivot Wheel:

1. The main axle for the pivot wheel starts as a vertical axle and a perpendicular axle joiner
2. Next, place an axel (try #4 or #5) through the round hole of the axle joiner. Then one or two SMALL wheels to the #4 axle. The wheels will attach tightly to the axle and the axle will spin easily in the joiner (round hole).
3. If bushings are needed to keep the wheels low enough and away from the robot body, add them to the vertical axle. Place a perpendicular axle 3L joiner where you want to connect the pivot to your robot.
4. Place two axle pins in the 3L joiner and connect to your robot as desired into the side of a horizontal beam. Place a bushing on the top of the 3L joiner to prevent your wheel from falling off when you pick up your robot.


1.

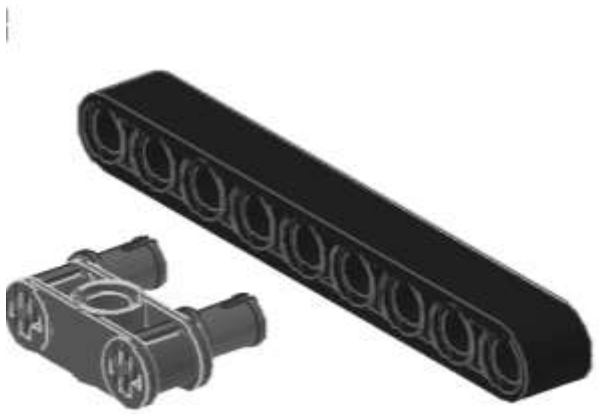

2.

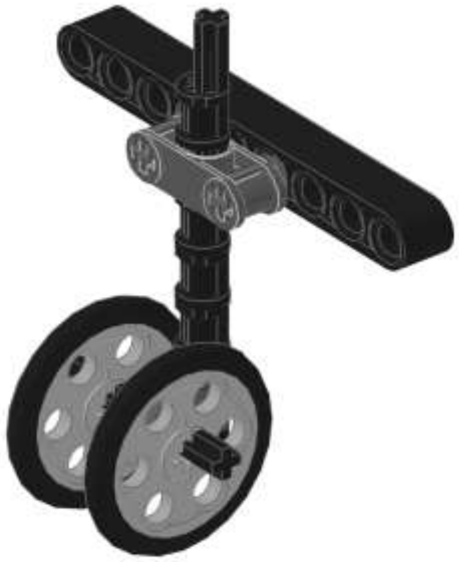

3.


4.

Attaching to the robot:

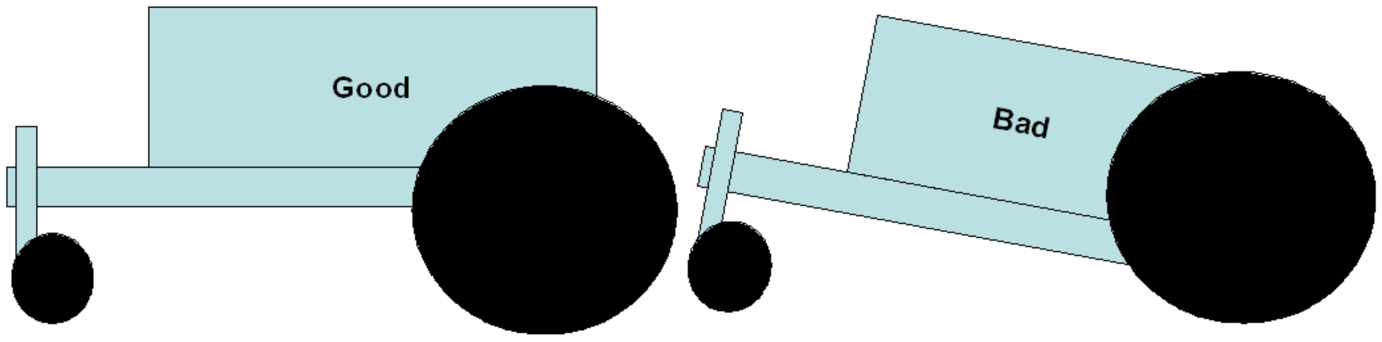
The easiest way to attach the pivot wheel is to snap in the axle pins (shown as blue in fig 4. above) into a **horizontal** beam. Studded or studless does not matter, as long as the beam has round holes in the side:


1.


2.

Problems with Pivot Wheels

1. The pivot wheel will only be as strong as the structure of the robot and the arm that connects the robot to the wheel. So build solid, connect as many knobs as you can and don't extend the pivot wheel too far out from the robot.
2. There must be enough distance for the pivot wheel to spin all the way around without hitting the robot body.
3. The pivot wheel has to be small so it doesn't have to be "dragged around" by the robot.
4. The pivot wheel should be level with the rest of the robot, see figures below.



Using Gears, Part I

Gears make all the difference on a robot or machine

They are used to **slow down** or **speed up** a moving drive system, or to make a drive system **more powerful**.

The Motors used by the NXT MindStorms kit are geared motors (they already have some gears inside them). Without any load on them they turn at approximately 170 revolutions per minute (RPM). The NXT rotational sensor is accurate to within 1 degree

When we say "no load" that means the motor is just turning, it isn't making gears, wheels, axles or other parts move as well. All of these things require additional energy and will slow down the RPM output by the motor. There is also a "no-load" for a robot, this is the difference between running a robot off the ground (wheels spinning) and on the ground (moving the robot).

There is also an UNGEARED 9 volt motor. These always need gears to be useful on a robot

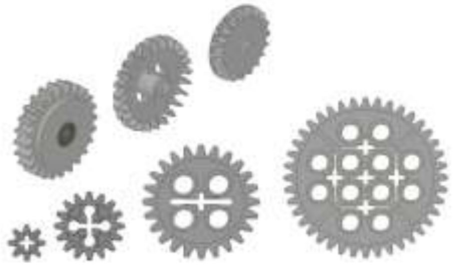
Types of Gears

The Gears used in the MindStorms kit include a number of sizes and types. The most common are:

- 8 tooth
- 16 tooth
- 24 tooth
- 40 tooth

Less common, but useful are:

- 24 tooth clutched gear
- 24 tooth crowned gear
- 20 tooth bevel gear



The last two are useful for changing axle direction and the 20 tooth bevel is used in the gear casing to make slip differentials.

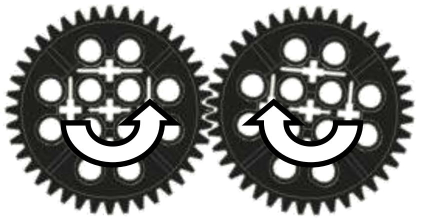
The clutch gear allows power to be transmitted from one gear to another but it STOPS this transmission (by slipping inside) so you won't damage the LEGO parts or motors.

All of the gears can be attached directly to the motor OR to an axle or connector with an axle end.

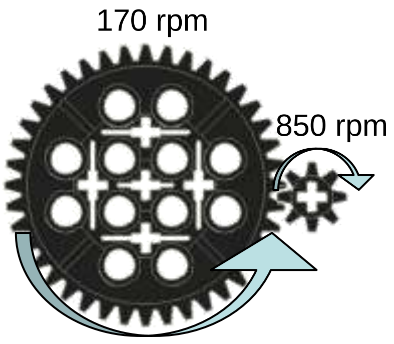
NOTE: LEGO® and MindStorms are the property of the LEGO Group

Gears can be used for a number of reasons:

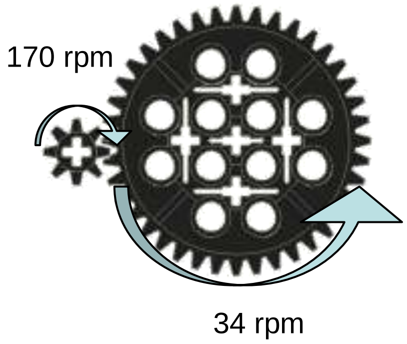
1. They can change the direction of rotation



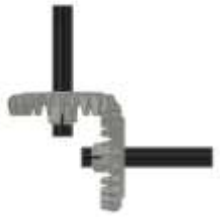
2. They can increase the speed (decrease torque)



3. They can decrease the speed (increase torque)



4. They can change the axis of rotation



Getting more power or torque out of your robot

There are times when you need a lot of torque. Maybe you are building a robot that can pull 25 kilograms across the floor and that old "big wheel directly on the motor axle" just isn't working any more.
You can get more power/torque for your system if you:

1. Add more motors
2. Decrease wheel size
3. Gear down your drive train.

If you "gear-down" you will move slower and have more torque to do the work that needs to be done.

Bulldozers, tractors, winches, and cranes are all examples of using gears to slow down while increasing the torque available

Getting more speed out of your robot

There are times when you don't want a lot of torque, but rather a lot of speed. There are a number of ways to speed up a robot.

1. Add more motors
2. Reduce the weight of the robot
3. Reduce the weight of the drive train (moving parts)
4. Increase the gear ratio of the drive system

Drills, high gears on bicycles and in cars, air plane engines/ propellers all need to spin at high speed.

Gear Ratio

The gear ratio of a system is the comparison of the drive gear (off the motor) to the end gear (attached to the wheel or other machine part).

1:1 Gear Ratio
This is when the first and last gears are the same size and spin at the same RPM

>1 Gear Ratio
This is when the last gear is moving at a higher RPM than the motor's drive gear (making the system go faster)

<1 Gear Ratio
This is when the last gear is moving at a lower RPM than the motor's drive gear (making the system go slower)

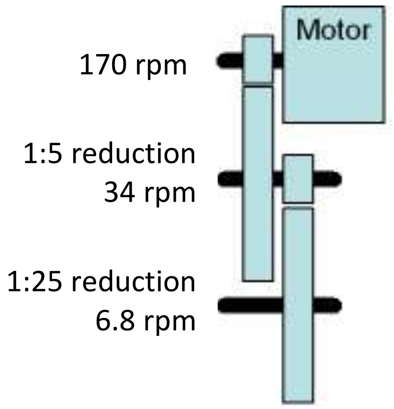
Gear Ratios are given as ratios. A gear ratio of 5:1 means that the last gear will be turning at 5 times the rate of the drive (motor) gear.

A gear ratio of 1:5 means that the last gear will be turning at 1/5th the rate of the drive gear.

Gear ratios can be calculated by comparing the number of teeth on the gears. So going from an 8 tooth gear to a 40 tooth gear gives you an 8:40 ratio (or simplify for a 1:5 ratio).

Making gear trains

A Gear train is a series of gears in a system that allows you to greatly increase or decrease the gear ratio and speed of the machine.



In this example we have one motor, two 8 tooth gears and two 40 tooth gears, plus the axles. The motor drive gear must turn 5 times before the 40 tooth gear turns once. For each time that 40 tooth gear turns the smaller 8 tooth gear on the same axle turns once (same axle right?). But that second small 8 tooth gear must turn 5 times before the 2nd larger 40 tooth gear turns once. The first gear reduction is 1:5 and the second is 1:5. We multiply them together and we get a total gear reduction of 1:25. The motor must turn around 25 times before the second large gear will turn around once.

Making stronger robot gear systems

Making stronger gear systems (not the gear ratio, just the basic mechanical engineering) requires that all axles and gears be placed in a strong manner.

Here we see each axle for each gear held in place by holes on either side. There is also just enough room for the gear but not enough for the gear to move around (side to side). Plates are used to hold the beams in place.

Remember: You can have speed OR torque, but not both.

MAINE ROBOTICS

Using Gears, Part II

5

Gear Spacing:

You can't always stick two gears next to each other. The distance between holes on the beams (the usual place to put axles) is set. You can change the vertical distance by putting plates in as spacers.

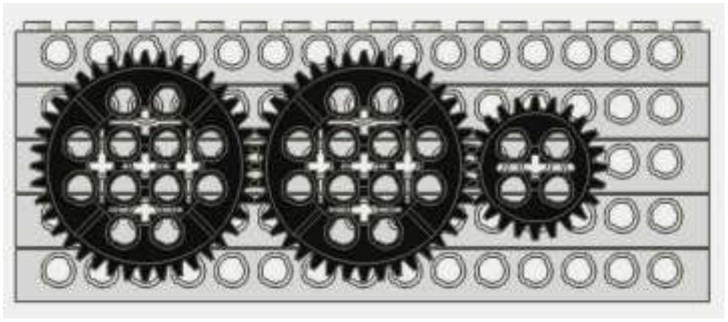
The following lists tell you which gears can be put next to each other and how the holes would align.

To read the charts, the first pair of numbers tells you which gears are involved. 40:24 would mean you are trying to put a 40 and 24 tooth gear next to each other. The two gears would have to be 4 holes apart on the beam.

Horizontal:
(Distance between centers of gears)

40:40	5 holes
40:24	4 holes
40:8	3 holes
24:24	3 holes
24:8	2 holes
16:16	2 holes
8:8	1 hole

NOTE: You cannot go directly from a 40, 24 or 8 tooth gear to a 16 tooth gear (horizontally).

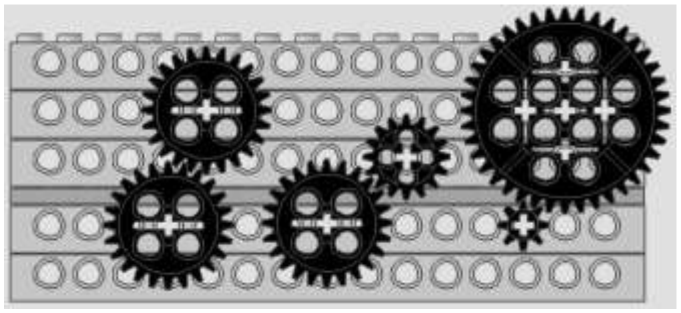

Shown here are 40:40:24 gears spaced horizontally with a spacing of 5 holes between the 40:40 and 4 holes between the 40:24 gears.

Vertical (beams with holes and no plates)

40:40	up 4	over 1*
40:24	up 3	over 2*
40:16	up 3	
40:8	up 2	over 2
24:24	up 2	over 2
24:16	up 2	over 1
24:8	No	
16:16	No	
8:8	No	* means imperfect fit

Vertical with spacer (beams with holes and 1 plate) for distances up don't count the plate, just the beams.

40:40	up 3	over 3
40:24	up 3	
40:16	up 2	over 2
40:8	up 2	over 1
24:24	up 2	over 1
24:16	up 1	over 2
24:8	No	
16:16	No	
16:8	No	
8:8	No	


Shown here are some common gear spacings with a plate in between:

- 24:24 with one gear 2 up and 1 over from the lower gear
- 24:16 with one gear 1 up and 2 over from the other
- 40:8 with one gear 2 up and 1 over from the other

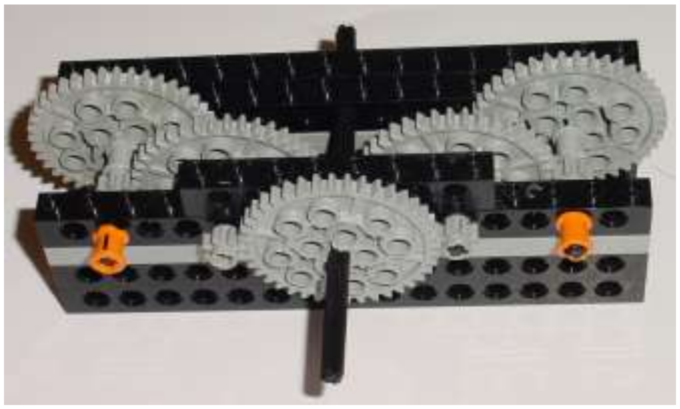
Vertical with 2 spacers (beams with holes and 2 plates) for distances up don't count the plate, just the beams.

40:40	No
40:24	No
40:16	No
40:8	No
24:24	No
24:16	No
24:8	up 1
16:16	up 1
16:8	No
8:8	No

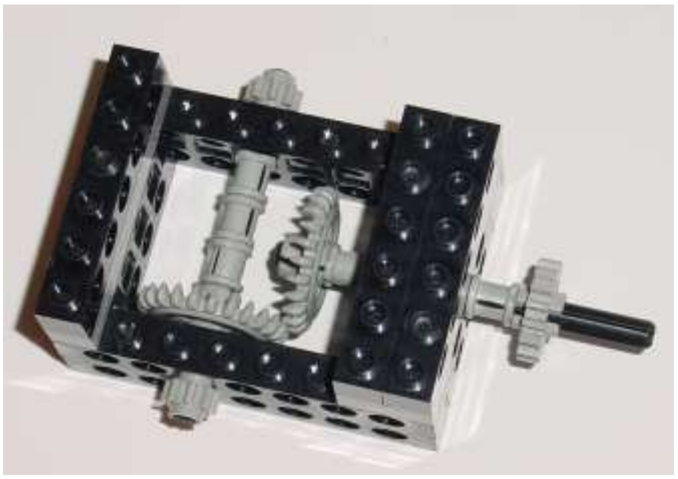
REMEMBER:

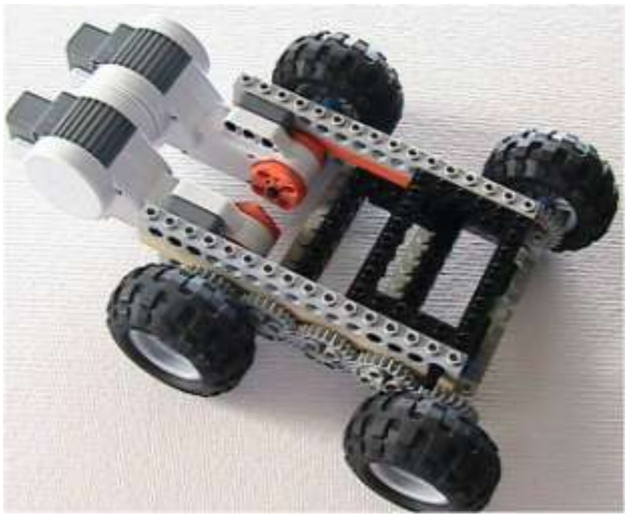
It doesn't matter if you go up or down or right or left. When a chart says "up 3" it just means the top gear is 3 higher than the lower gear. And "over 2" means either to the right or the left, it doesn't matter.

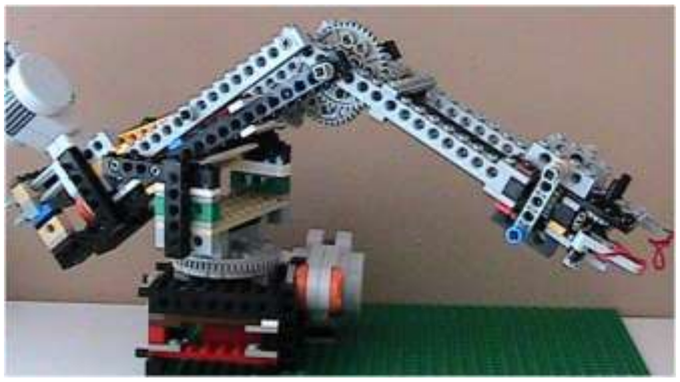
Some Sample Drive Trains:


This gear train came from a work horse robot. It has almost a 1:250 gear reduction. It moved extremely slow but could pull about 50 pounds across the floor. Notice how each of the side 40 tooth gears also run the 8 tooth gears on the same axle. These then turn the mid-right and mid-left 40 tooth gears, which are also attached to 8 tooth gears, which in turn make the center 40 tooth gear turn.

This part of the drive train brought the power from four motors (two each for the outer 40 tooth gears) through a 1:25 gear reduction to finally power the center axle. From the center axle additional drive train components lead to the front drive wheels and the rear drive wheels.


This is a simple right angle gear system. A drive shaft comes in on the right and is used to power an axle that is at a right angle to the first axle.


Here you can see a vehicle that was designed to be four wheel drive by using a set of 24 tooth gears.

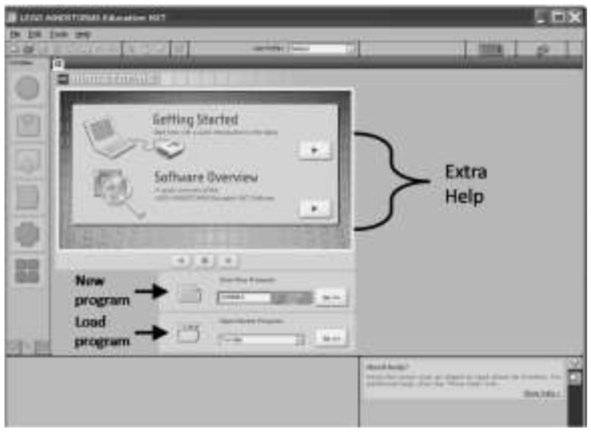

This robot uses a variety of gears and drive-trains to create a moving robotic arm that can pick up and sort objects (such as other LEGO pieces).

NXT Software Basics 6

The NXT Educational Software:

WHERE TO BEGIN?

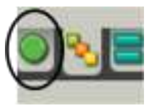
After you open the software, choose to either create a new program (don't forget to name it) or load a saved one and select 'go':



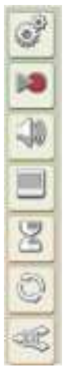
You can also go right to the help files from this opening screen (see 'Extra Help' in the fig. above).

THE COMMON PALETTE:

Before you start programming, be sure that you have selected the SMALL green circle in the tab on the lower left hand side of your screen.



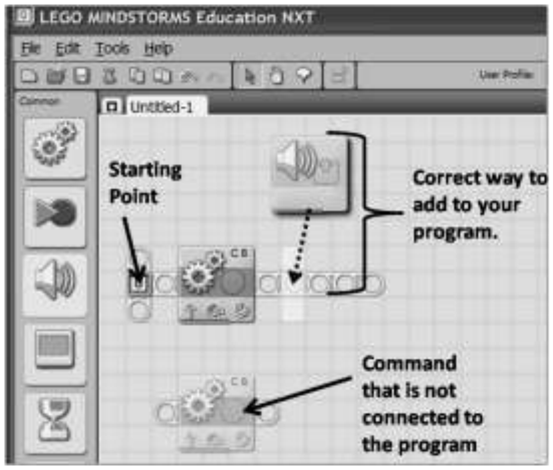
Once you have selected this tab, the left hand side of the screen should look like the figure on the right →



From this tab, you can do nearly everything you will need to do with your robot. From top to bottom, you can control the motors, record motions, play a sound, display on the NXT screen, wait, loop, or switch.

BUILDING A PROGRAM:

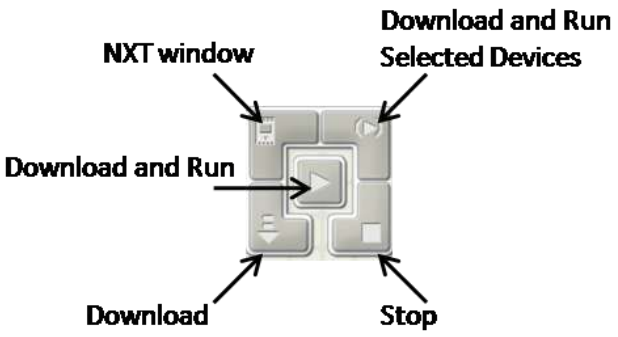
When you are creating a program, you will be selecting different commands on the left side of the screen, and dragging them onto your program surface. If you want to add a command to your program, you must make sure it is connected to something that eventually leads back to the starting point. If your command is not connected properly, the colors will appear faded:


LEGO MINDSTORMS Education NXT
File Edit Tools Help
Untitled-1
Starting Point
Correct way to add to your program.
Command that is not connected to the program

NOTE: The NXT software reads commands sequentially from left to right, so order matters when programming.

DOWNLOAD & RUN:

When your program is ready for testing (make sure to save it first!), you have to upload it onto the NXT. Whether you have Bluetooth capabilities or not, you can upload to your NXT via the set of icons in the lower right side of the window:



The “NXT Window” and “Download and Run Selected Devices” are more useful when you are using multiple Bluetooth devices. If you only have one device and it is not Bluetooth, you need to check that the USB cable is connected between your computer and NXT. Be sure that your NXT is ON and begin downloading.

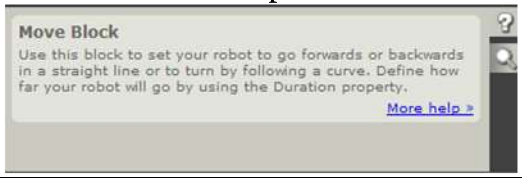
If your NXT is connected into the computer, and you want to download and then immediately run the program, select the middle “Download and Run” icon. If you wish to quit your program while it is running, you can either press stop (the NXT must still be connected to your computer) or the “back” button on the NXT. If you do not wish the program to run immediately after download then just select the plain “Download” icon.

ADDITIONAL HELP:

There are two forms of help files in the MINDSTORMS programming software: One uses only pictures and video while the other one uses pictures and text. To access the picture only help file, look to the upper right hand corner and click on the brick tab. This will drop down the help menu as shown:

To access the picture and text help file, just go to the upper left hand corner and go to “help → contents and index”.

Additionally, you can mouse over any piece of your program at any time and the lower right hand corner will display some information about that piece:


Move Block
Use this block to set your robot to go forwards or backwards in a straight line or to turn by following a curve. Define how far your robot will go by using the Duration property.
More help »

OTHER FUNCTIONS:

Here is a list of other icons you may find in the software and each of the functions they perform:

The **Pointer Tool** is used to select or move programming blocks. You can find it in the upper section of the software window

The **Pan Tool** can be used to click and drag to a new area of the program (useful if your program gets long). Look for this one next to the pointer tool.

The **Comment Tool** is used to write text into your program for future reference. This text does not affect the program in any way. It is simply meant to be used to help you keep track of all the parts of your program. It can be found next to the pan tool.

Create My Block will take a section of your program (which you have selected) and save it as a special block. This block is useful when you have a line of code you wish to repeat again or to make a large program easier to see.

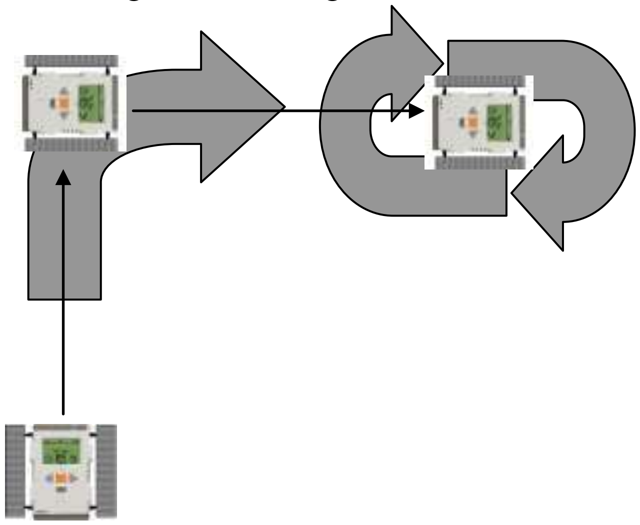
Turning Your Robot

7

The Mission: Dead Reckoning

Build a robot and program it to travel forward for 2 feet; stop; turn 90º to the right; stop; go straight forward for 3 feet; stop; turn completely in a circle; stop; end the program.

This form of navigation (without any FEEDBACK from the environment) is called DEAD RECKONING. Navigation using the position of the stars, the Sun, the moon, etc. was called "Live reckoning" and working without these feedback methods was dead reckoning.



What you will need:

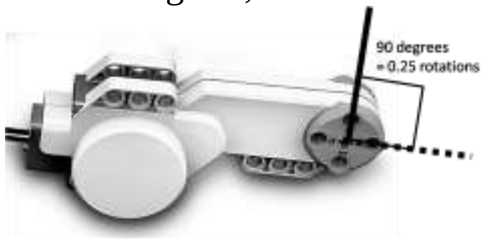
1. A robot (either wheeled or treaded)
2. If wheeled, then you need a pivot wheel or wheels to allow the robot to turn. See module 2 for more information on pivot wheels (think of the front shopping cart wheels, they spin around).

Figure 1: This example shows a treaded robot, but it could be a wheeled robot as well.

BIGGEST PROBLEMS:

1. Getting a good turn.
   a. If you have a treaded robot, turning is easy.
   b. If you have a wheeled robot, you have to have a pivot wheel, or you can use skid plates (small round disks that you can place on the bottom of your robot)

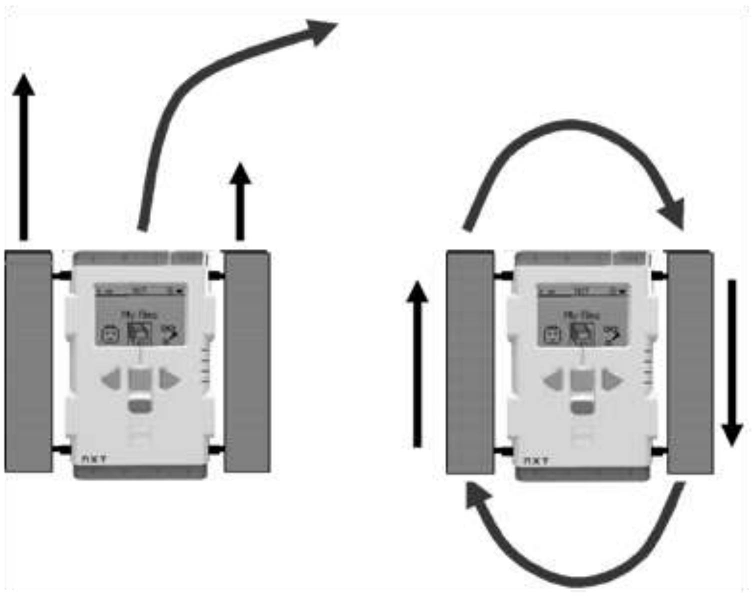
2. Learning to program in fractions of a second and fractions of a rotation.
   a. Since we want the robot to turn as close to 90º and later in a 360º circle, we will need to do TRIAL AND ERROR for turning. Each robot will behave differently so you MUST try something out, then adjust the settings that you can control (called VARIABLES). **Remember:** rotations and degrees are for the AXLE, not the robot. 1 rotation = 360 degrees, and thus 0.25 rotations = 90 degrees.



Turning the Robot:

Robots are usually turned by one of two methods.

1. Run one motor at a higher power level/speed than the other. This causes a forward turn.
2. Run one motor forward and the other motor backward. This will cause the robot to turn in place (more or less depending on the robot)



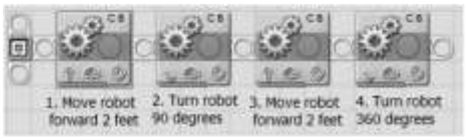
The straight arrows represent the direction of the treads (or wheels) and the size of the arrow represents what the power is set at. The curved arrows show how the robot will turn.

In the first example both motors are running forward and the robot travels forward to the right.

In the second example the motors are reversed and the robot "spins in place". If this was a wheeled robot it might not spin exactly in place, but its close.

Programming the Robot:

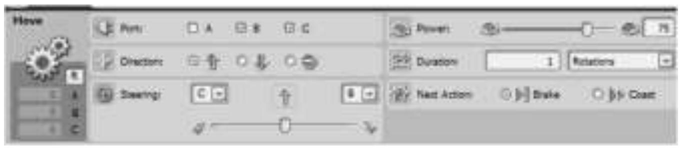
The new NXT software makes programming a robot really easy. First, be sure your motors are plugged into ports B and C of your NXT. We want to make our robot travel two feet, turn 90 degrees, travel two more feet, and then turn 360 degrees. All we need to do this are four movement commands:


1. Move robot forward 2 feet
2. Turn robot 90 degrees
3. Move robot forward 2 feet
4. Turn robot 360 degrees

Since you can only program your motors to be on for a certain amount of time or rotations, you will have to find out by **trial and error** how to turn your robot 90 degrees. We recommend you record the amount of rotations it takes, not the amount of time your motors are on, to turn 90 degrees. This is because as your batteries drain, your motors will travel slower and thus your time required to make a right angle will increase.

To make a turning command, open up the common palate (green circle) and select the picture of the gears ("Move").

Next, connect the move command onto your program and observe the bottom-left hand corner when you highlight it. There are many options you have for adjusting the move block:



To steer your robot, you want to adjust the **steering** and **duration** options. As always, make sure your ports from your programming block match how you connected them on the NXT. Depending on which way you want your robot to turn (it also depends on the orientation of your motors), you will need to move the steering bar to the right or left. For the sharpest turning, move the steering bar completely to the right or left.

Finally, adjust the duration option to determine how to get your robot turning exactly 90 degrees (trial and error).

Using A Touch Sensor

8

The Mission: Use a Touch Sensor to provide feedback to the robot

Build a robot that will drive forward until it hits a wall; then backs up for 3 feet and stops the motors; then stops the program.

What you will need:

1. A robot (either wheeled or treaded)
2. A touch sensor attached to Port 1
3. A way to attach the touch sensor to the front of the robot
4. Some kind of arm or mechanism that pushes the touch sensor button whenever the front of the WHOLE robot is touched.

BIGGEST PROBLEM:

Figuring out the answer to #4 above…

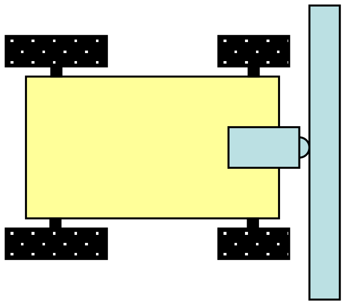
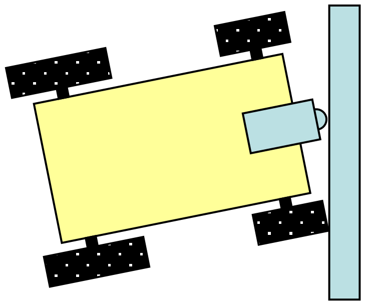
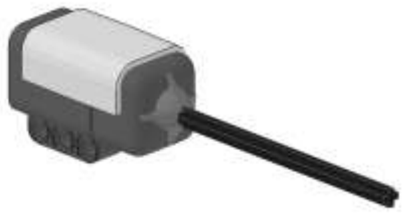




Figure 1: Robot hits wall directly and touch sensor works
Figure 2: Robot hits wall at angle and touch sensor isn't triggered.
Figure 3: The NXT touch sensor can have an axle extend its reach.

Possible Solutions:

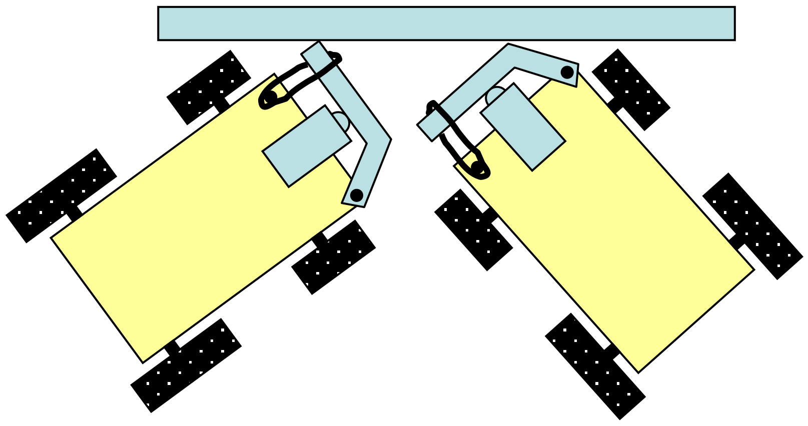


Figure 3: This robot has a pivoting arm in front of the sensor with an elastic gently holding it in place. When the arm hits something it pivots back activating the sensor. Note also that the front wheels got moved back so

they wouldn't interfere with the sensor. **USE YOUR IMAGINATION TO COME UP WITH A SOLUTION!**

Programming the Robot:

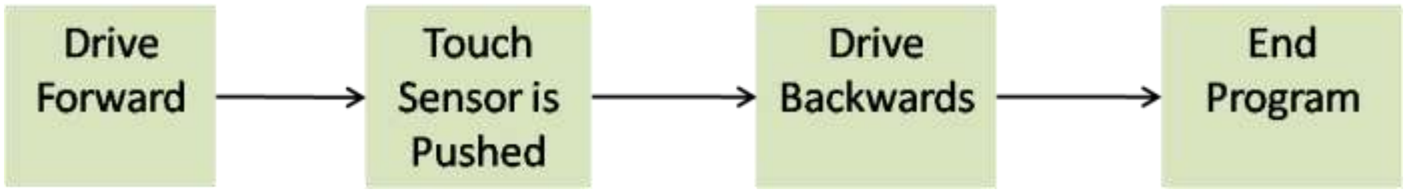


Figure 4: This is a flowchart showing what we want to do with the program. It is always good to think out what we want to do ahead of time.

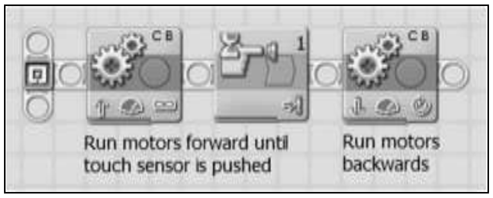
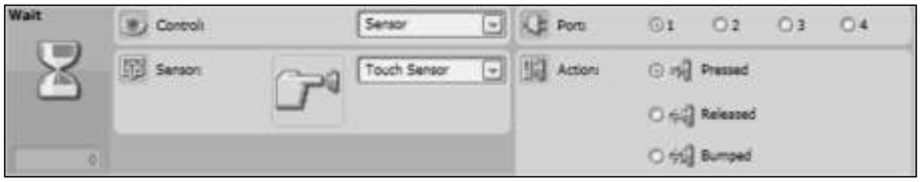


Figure 5: Is an example of a MINSTORMS program that can be used to accomplish the mission. Remember to select the correct ports for both the move blocks and the sensor block when programming.

Things to Remember:

The touch sensor block has three types of triggering options:



Wait

Control: Sensor

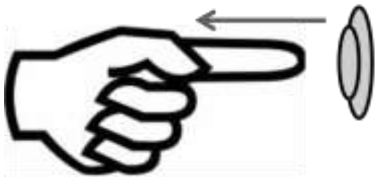
Sensor: Touch Sensor

Port: 1 2 3 4

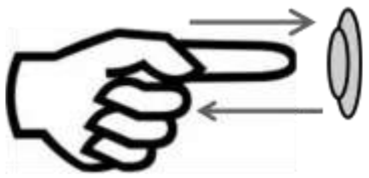
Action: Pressed / Released / Bumped

1. **Pressed**: Use this option when you wish to have the program react when the touch sensor is pushed in. For this mission, you will probably want to use this option.

2. **Released**: Selecting this option means that the program will wait until the touch sensor is <u>not</u> pushed in or is released.



3. **Bumped:** This option will require the program to wait for the sensor to be pushed in and released again before continuing.



Using a Light Sensor

The Light Sensor

The light sensor is one of the basic sensors that are available for use with robots and for collecting data.

The light sensor will read a value of from 1 to 100 with 1 being very dark and 100 being extremely bright. The sensor can work in two ways.

1. It can read the amount of **ambient** or surrounding light (imagine holding the light sensor up in the air and measuring how much light there is in a shady or sunning area.
2. It can read the amount of light **reflected** back to the light sensor.

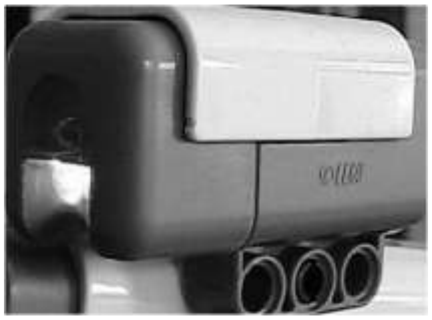


Figure 1: LEDs on a light sensor

Either way there are two **LED**s (Light Emitting Diodes) on the sensor as shown in figure 1. One emits light and one is used to determine how much light is received by the sensor. The neat thing about LEDs is that if you run electricity through them they produce light and if you shine light on them they produce electricity. So the NXT is measuring how much electricity is being produced by the **receiving** LED.

When you use the sensor to measure changes on a surface as you would on a floor, table top, or mat you are measuring how much light is reflected back. A dark area reflects less than a light area.

Sometimes it is very important how you place a sensor. The closer it is to the surface or object being checked the more accurate and repeatable the reading. So if you are reading off a floor or mat, make sure the sensor is close to the surface.

Attaching the Light Sensor

The light sensor must be attached with a cable to any of the four sensor ports (1,2,3, or 4).

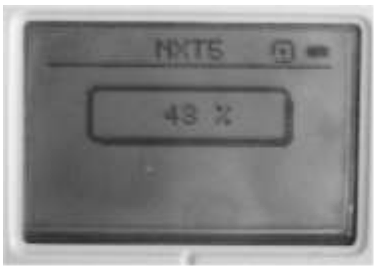
For the light sensor, the NXT program will look for the sensor on Port 3 **UNLESS** you have programmed it to look on a different input port.

Figure 2: light sensor on NXT

USE THE VIEW OPTION

You can get a light reading without having to write a program. This is very useful in determining what values to set in your program.

Turn your NXT on by pushing the orange button. Be sure your light sensor is hooked into a port. Arrow over and select “view”, then scroll over again until you reach “reflected light” or “ambient light”. Pick your desired method of measuring and then select the port that you have plugged your sensor into, and you should arrive at a screen like this:



NXT5

43 %

This is the current reading from your sensor. Point the sensor close to different objects and see what it says! Don’t forget to experiment with both ambient and reflected modes.

Some Special Problems:

- The light sensor is sensitive to light, so the better shaded and closer to the ground, the better it works. It should be within a ¼ of an inch and pointing directly down. If it is too far above the ground it will not be very accurate.
- If you are trying to point the light sensor at the floor, try to design a mount for the sensor so that it rides close to the ground (without touching it) to get a good light reading:

- This sensor default is to attach it to SENSOR PORT 3. If you attach it to sensor port 1, 2, or 4 you have to tell the program that is where you put the sensor. This can be done by simply selecting the desired port in the bottom of your programming window after you have selected the light sensor using the pointer tool.

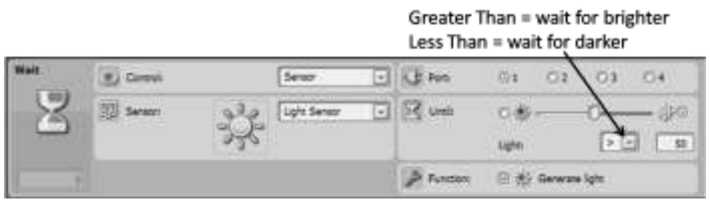
Sample Missions:

1. Program a robot to move forward across the floor or mat until it reaches a dark line. Then stop.
2. Program a robot to move forward across the floor until it reaches a dark line, reverse and come back until it reaches a second dark line.

Programming using Light Sensors:

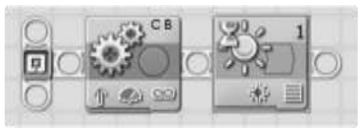
Within the basic light sensor commands, there are:

1. **Wait for Brighter**: This will wait until the sensor reads a value that is **greater than** a selected reference value (light > $\boxed{\text{xx}}$).
2. **Wait for Darker**. This will wait until the sensor reads a value that is **less than** a selected reference value (light > $\boxed{\text{xx}}$)



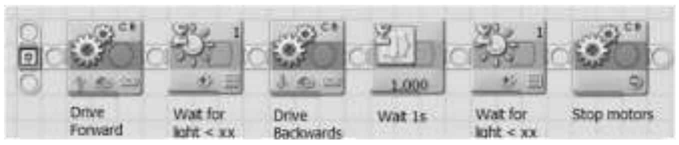
When using these functions, you must FIRST determine a reference point based on your tests when using the view mode of your NXT (see attaching the sensor on the other side).

Sample program 1:



Shown here is one possible program for the first sample mission. The motors are turned on and set to run indefinitely ("Unlimited"), and then we wait for a light value that is less than our chosen value. Once we see the darker value of the line we end the program.

Sample program 2:



In the second example mission we want to be a little more complicated and the steps are listed here for the sample program above.

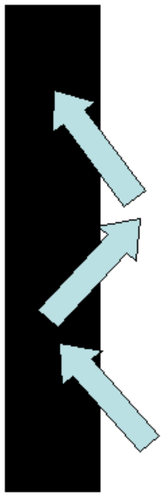
1. Start the program
2. Run the motors forward
3. Wait until we see the dark line
4. Run the motors backward
5. Wait 1 second (this is to clear the first line, it is very important that you are not still on the dark line or you will never be able to find a "darker" value as you go looking for the second line)
6. Wait for darker (the second line back at the starting point)
7. End Program

Line Following 1

Building a Line Following Robot:

A robot that can follow a line has more control over its movement than one that has to move by what is called "dead reckoning". When your robot relies only on timers to know how far it has gone and how much it has turned it is dead reckoning, and it could be dead wrong.

When a robot uses sensors it is getting feedback from its environment so it can "sense" where it is. This is called "live reckoning", and is often slower than dead reckoning, but usually more accurate.



A line following robot will really follow the EDGE of a line. When the light sensor is over the lighter color it will turn one way, and when over the darker color turn the opposite direction. This way it keeps zigzagging across the edge of the line as it moves forward.

Some Special Problems:

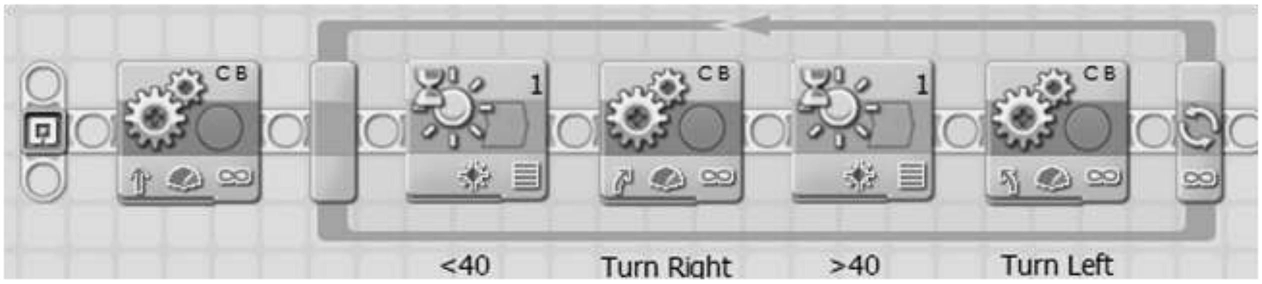
The light sensor is sensitive to light, so the better shaded and closer to the ground, the better it works. It should be within a several millimeters (¼ inch) and pointing directly down. If it is too far above the ground it will not be very accurate and will be affected by room light. The sensor values you set will depend on how much light is in the room and how dark or light the surface and line are. Use the VIEW option on the NXT to find out what values you have and then pick something in between. For example, if the floor is 45-50 and the line is 25-30, then set the comparison to >37 (halfway between). This is probably the most frequent problem builders have.

The second most common error is to attach the sensor to one port and program it for a different port. The light sensor default is to attach it to SENSOR PORT 3. If you attach it to sensor port 1, 2 or 4, you will have to tell the program that is where you put the sensor by modifying the settings of the "Wait For" block.

SAMPLE MISSIONS:

1. Build and program a robot that can move forward while following a black line.
   This should work even if the line turns. Work at getting the robot under control, this is not a time for faster is better.
2. Follow a line until the robot hits an object (wall) and then stop (two sensors and harder).
   Here we are using 2 sensors so make sure they are programmed onto two different sensor ports and that the program has them listed on two different ports.

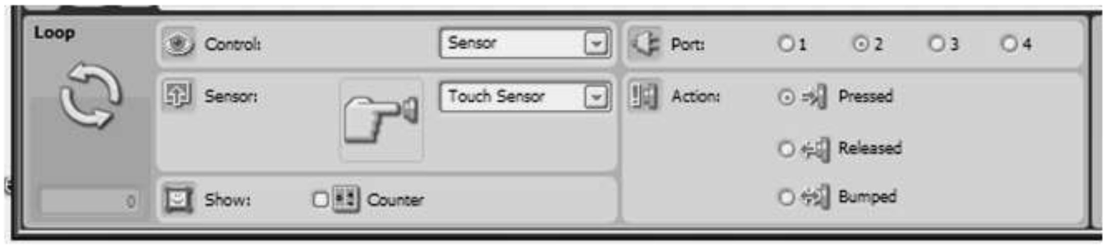
Program 1: Follow a line



1. Start the program (with the light sensor over the darker area)
2. Run motors forward (your robot may be different to go forward or may need to adjust the speeds)
3. Form a loop. This will allow your robot to continue line following as long as you like.
4. Wait for a light value.
5. Once you have "seen" the light, make a "move" block and adjust the steering bar to turn left
6. Wait for a dark value. This is when you've swung back over the dark line.
7. Once you have "seen" the dark, make a "move" block that will turn right.
8. Make sure everything, except the initial run motors forward, is contained within the loop.
9. You can program your motors to stop by requiring a set amount of loops or time rather than indefinitely, as shown above.

Program 2: Follow a line and stop at the wall

For the second example, all you have to do is adjust the settings of your loop in program 1 so that the loop will end when the touch sensor is triggered. To do this, select the loop by clicking on either the left or the right hand edge of the loop. Look down in the lower left of your window and change the "controls" option from "forever" to "sensor":



Loop

Control: Sensor

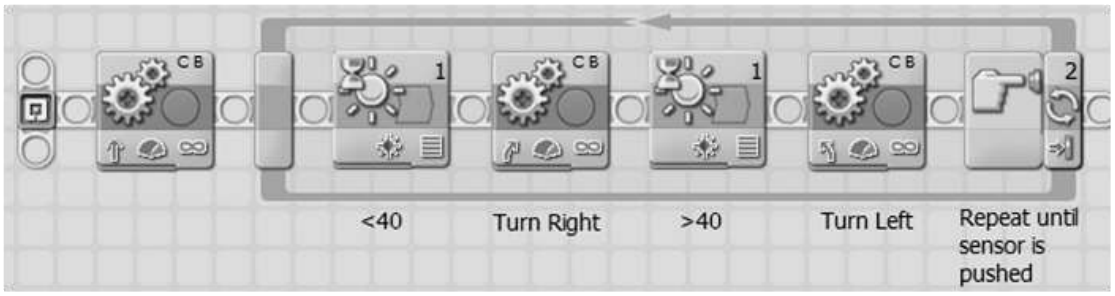
Sensor: Touch Sensor

Show: Counter

Port: 1 [✓ 2] 3 4

Action: [✓ Pressed] Released Bumped

When you do this, your loop should be slightly modified and look something like this:



Your robot will now follow the line until the touch sensor is triggered. Make sure the ports for each of the sensors are the same as the ones you set in the program otherwise your robot will not function properly!

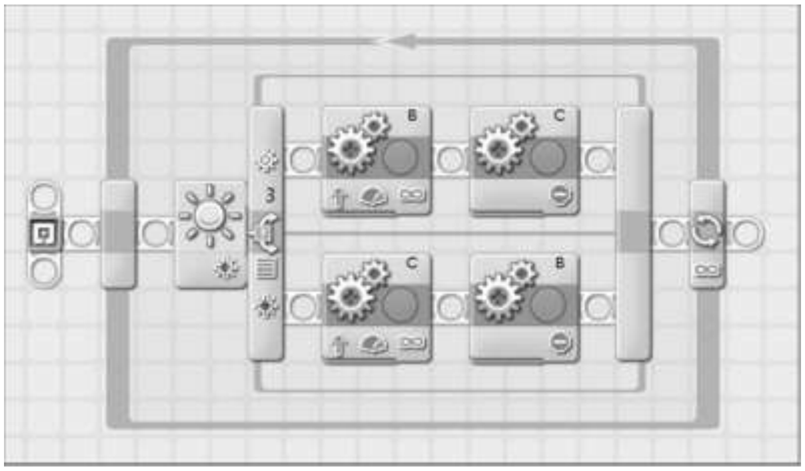
Line Following 2

10a

Building a Line Following Robot:

See Line Follower 1 for directions on setting up a robot and attaching the light sensor.

Program 1: Follow a line using a Switch



1. Start the program (with the light sensor over the darker area)
2. Use a LIGHT SENSOR switch inside of a LOOP. Each time the program loops it will ask the question, “do I see the line (dark) or the floor (light)?” and then go right or left depending on your answer.
3. Notice we only have ONE motor running while the other is turned off. So when B runs C is turned off, and visa versa.
4. Notice also that we have the duration for the motors set to UNLIMITED. This is the ONLY setting that allows the program to continue. If you try to use timed or distanced travel, then the program waits their for the command to finish, then it will ask the question again.
5. So this program repeatedly is asking the question, “do I see the line or the floor?”
6. Experiment with different options, such as turning the robot, rather than stopping one of the motors. As shown below: