

# Rescue Rover

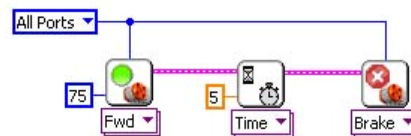
## Overview

In this challenge students will be presented with a real world rescue scenario. The students will need to design and build a prototype of an autonomous vehicle to drive through a hazardous environment to perform a rescue operation. The students will prototype a rescue vehicle using the LEGO MINDSTORMS® NXT kit and program the NXT brick using LabVIEW Education Edition (LVEE). The students will also be introduced to the engineering design process in this lesson.

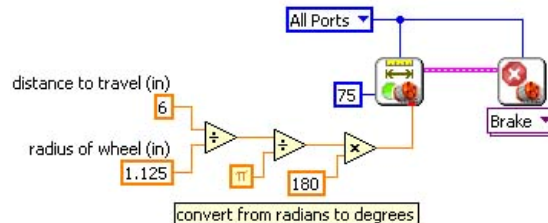
Sample Vehicle  
(without supply carrier)



Sample Program #1



Sample Program #2



Expectations	Evidence
<p>Students should be able to:</p> <ul style="list-style-type: none"> <li>• Build a sturdy NXT car with 2 motors using the NXT kit.</li> <li>• Use LabVIEW to program their NXT vehicle to drive a specified distance first using direct mode, then remote mode.</li> <li>• Use the engineering design process to solve the given challenge.</li> </ul>	<p>Evidence of learning found in:</p> <ul style="list-style-type: none"> <li>• A NXT vehicle that is sturdy, within size requirements, and able to drive a specified distance using direct and remote mode.</li> <li>• Commented LabVIEW code.</li> <li>• Engineer's journal.</li> </ul>

# Lesson 1

# Rescue Rover

## Suggested Time

90 minute

## Vocabulary

(See Appendix L)

Autonomous vehicle  
Virtual Instrument (VI)  
Front panel  
Block diagram  
Code  
Direct mode  
Remote mode  
Compile  
Download  
Engineering design process  
Prototype

## Materials

### Each student:

- Engineer's Journal

### Each student group:

- LEGO MINDSTORMS kit
- Computer with LabVIEW Education Edition

## Teacher Preparation

- Build a sturdy car example.
- Arrange students in pairs.
- Distribute Engineer's Journals.

## Challenge

A group of mountain climbers are stranded atop Mt. Everest. Unfortunately, a storm is approaching and rescuers will not be able to attempt a rescue until the storm has passed. The stranded mountaineers are in desperate need of food, water, and medical supplies. The conditions are too harsh for a helicopter to drop off the supplies.

Your team of robotics specialists has been brought in to build an **autonomous vehicle**, no larger than 12" x 9" x 9" in dimension, that can deliver the much needed food and medical supplies to the stranded mountain climbers. In the first part of the challenge you will build a small prototype of the vehicle that can drive up a straight, narrow path. You have one hour to complete the challenge.

## Background

This lesson introduces the students to the LEGO MINDSTORMS NXT construction kit, LabVIEW Education Edition software, and the engineering design process.

The NXT kit contains an NXT programmable brick, technic building pieces, motors, and the following sensors:

- Light sensor
- Touch sensor
- Ultrasonic sensor
- Sound sensor
- Rotation sensor (built into the motors)



The students can build a wide range of robotic devices using this kit.

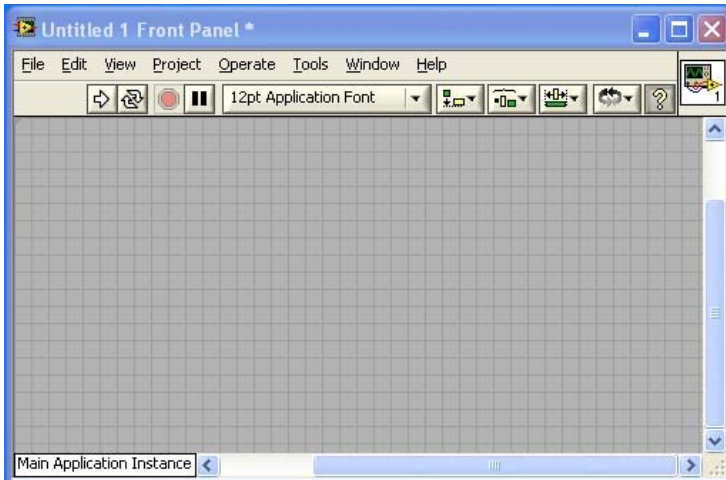
## Lesson 1

## Rescue Rover

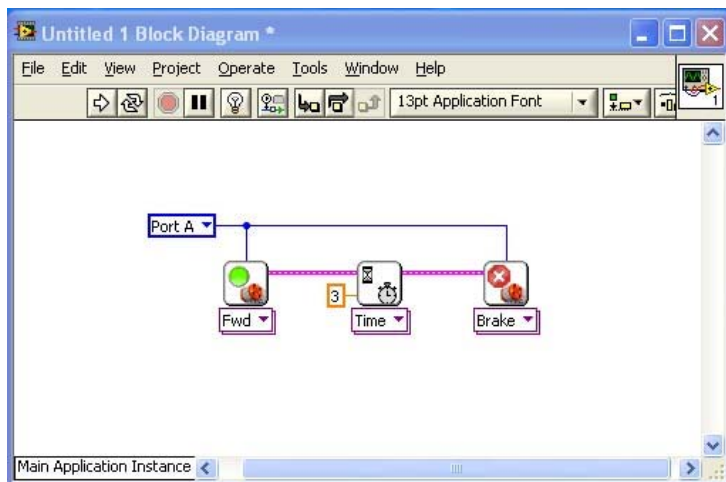
### Real World Connection

Engineers in industry use LabVIEW programming software to set up tests, collect data, etc.

The NXT brick can be programmed using LabVIEW Education Edition, a graphical programming language from National Instruments. LabVIEW programs are called “**virtual instruments**” (VI) and every program consists of a front panel and a block diagram as shown below.



Front panel



Block diagram

The **front panel** is where you create a user interface and the **block diagram** is where you write your code by wiring together graphical representation of functions and VIs. The **code** is a series of commands that tells your robot what to do. The code in the above block diagram turns on the motor attached to port A for 3 seconds then stops the motor using the brake.

## Lesson 1

## Rescue Rover

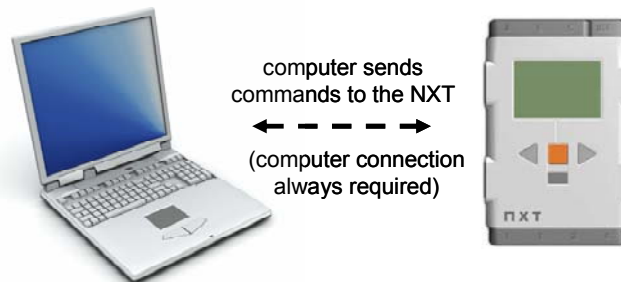
### Programming Tip

You can think of direct mode as the computer being the “brain” and remote mode as the NXT brick being the “brain”.

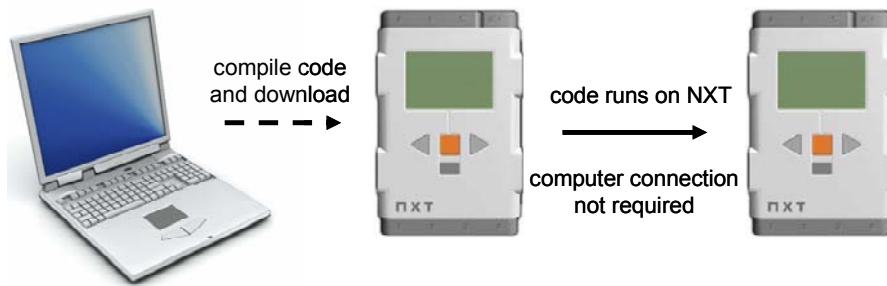
Programs in remote mode take less time to execute than programs in direct mode because information does not need to be sent back and forth between the computer and the NXT brick. For this reason you will want to execute most of your programs in remote mode.

Direct mode is useful for debugging a program because you can display output values to the front panel. You can also plot sensor values on the front panel in real time.

LabVIEW programs can be executed in two modes: direct and remote. In **direct mode**, the NXT brick is connected to the computer via USB or Bluetooth and the program executes on the computer. In **remote mode**, the program created on the computer is **compiled** and then **downloaded** to the NXT brick, and then the program can be executed from the NXT brick any time--the NXT brick does not need to be connected to the computer.



Direct mode



Remote mode

This lesson covers both direct and remote mode programming.

The **engineering design process** is an eight-step guideline that engineers follow to ensure that their product is designed efficiently and effectively (see Appendix A). The students will need to develop possible solutions to address the mining challenge, then select the best possible solution and construct a prototype of the rescue vehicle. Vehicles that are not sturdy, unable to drive, or does not meet the design requirements will need to be redesigned.

## Lesson 1

## Rescue Rover

### Real World Connection

In a real world rescue mission, engineers would have limited time, materials, and space to develop a solution. Explain to the students that the challenge is intended to simulate the real world so they all need to think and act like engineers.

### Hardware Tip

Note that a rotation sensor is built into the motors. It is the only sensor that does not plug into ports 1, 2, 3, or 4.

### Classroom Tip

Having a pre-built example for students is a great way to introduce a project. This allows students to gain ideas and more clearly understand the challenge.

## Instructions

### Part I: Introduction

**10 minutes**

1. Introduce the challenge to the students. As a class, identify the problem and design constraints of constructing the model (step 1 of the design process).
2. Define a **prototype** as a model of a design on which tests can be performed to evaluate whether the design should be used for the final product.
3. Introduce the students to the NXT hardware that they will use to construct their rescue vehicles.
  - a. Go over the names of the parts in the NXT kit (see Appendix B).
  - b. Explain that the NXT brick has 3 output ports (A, B, C) on the top where motors are attached and 4 input ports (1, 2, 3, 4) on the bottom where sensors are attached.



- c. Show the students the sample sturdy car that they will build. Students can either follow directions to build the car or design a car that look completely different, but still meet the size and structure requirements.

## Lesson 1

## Rescue Rover

### Building Tip

Visit [LEGOengineering.com](http://LEGOengineering.com) and select the Building & Programming Guide tab for more examples of NXT cars and building tips.

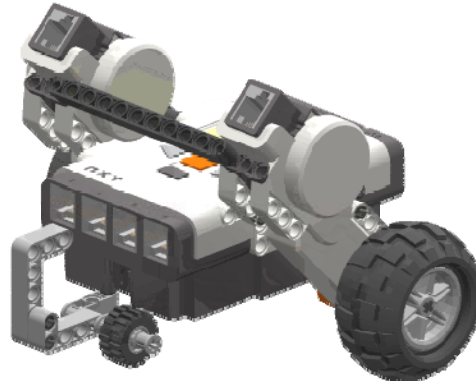
### Why?

An important part of an engineer's job is to document his/her designs, so that other engineers can add features, make changes, and/or use the design to gain ideas on future designs. Students should get in the habit recording their work.

### Part II: Building

**30 minutes**

1. Give each pair of students an NXT kit and instructions for how to build a basic car show below (see Appendix C).



2. Have the students build a two motor car either following the building instructions or one of their own designs.
3. When the students finish building their cars, they should spend a few minutes brainstorming how to build a container that can carry supplies (represented by a red and a blue ball that comes with the kit) to the stranded mountain climbers. The container can be a trailer and/or a box on top of the NXT car. The students should write or draw their ideas in their Engineer's Journal. Remind the students that the entire rescue vehicle, including the supplies container, needs to be smaller than 12" in length, 9" in width and height.
4. Have the students build the container and attach it to their car.
5. The rescue vehicle (car plus container) has to pass the following tests to be deemed sturdy:
  - a. Shake test –The teacher must be able to shake the car without pieces falling off.
  - b. Drop test – The teacher must be able to drop the car from the height of his/her ankle without the car breaking.
6. Rescue vehicles that are not sturdy need to be redesigned.
7. As the students finish building their rescue vehicles, remind them to draw diagrams of their finished designs in their Engineer's Journal.
8. Have the students set the rescue vehicle aside as you introduce the programming portion of the activity.

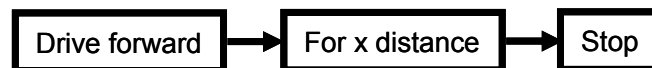
**Programming Tip**

Ask the students to think of “programming sequence” commands they would give themselves to complete the task.

You can compare palettes to toolboxes. There are different palettes or toolboxes that hold different VIs or tools to create your own program.

**Part III: Programming****30 minutes**

1. Explain to the students that they will now program their NXT rescue vehicles to operate autonomously—without input from the user. A program is a series of commands that the NXT follows. These commands need to be as specific as possible. As a class try writing out a series of commands the rescue vehicle needs to follow in order to succeed in the mission. For example:



2. Introduce the students to LabVIEW Education Edition software. Explain the basics of LabVIEW by giving the students a tour of the core components (front panel, block diagram, palettes, etc.). Most of the VIs they will want to use in this activity, like motor or wait commands can be found in either the NXT I/O or Complete palette shown below.



NXT I/O palette

## Lesson 1

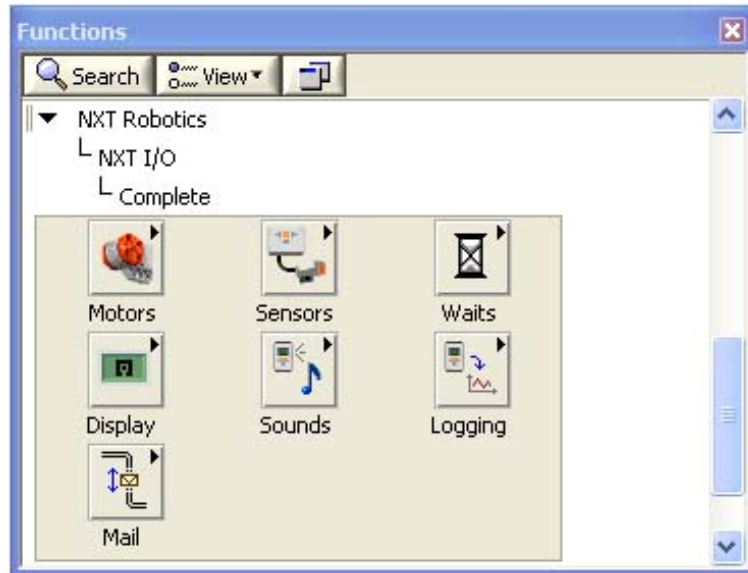
## Rescue Rover

### Programming Tip

The NXT I/O palette contains most, but not all of the functions found in the Complete palette. The difference between the two palettes is that VIs on the NXT I/O palette has polymorphic VI selector menus (purple drop-down menu) that allow you to select a few different modes of the VI without physically replacing the VI.

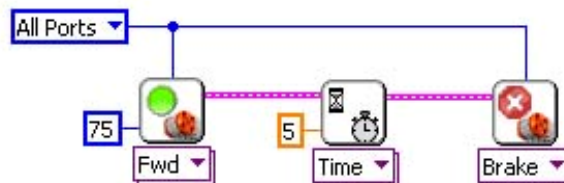
Turn on context help by selecting "Show Context Help" from the Help menu.

NXT Module Help is also found under the Help menu.



Complete palette

3. Show the students where they can find help resources such as context help and the detailed help documentation for the NXT Module. Context help describes what the VI does, lists the available inputs and outputs, and the default input values. The detailed help documentation includes tips, a sample program, and examples.
4. The first program will be written in direct mode. Demonstrate how to program the NXT brick using LabVIEW by writing a short program to drive, wait x seconds, then brake as show below.





## Lesson 1

## Rescue Rover

### Programming Tip

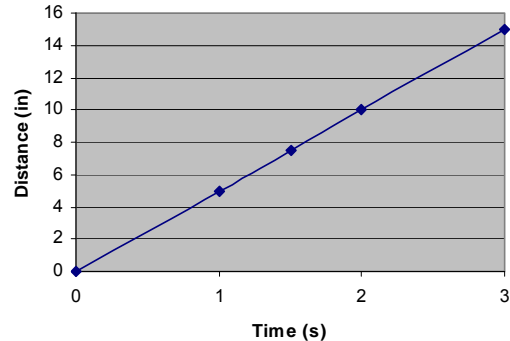
All VIs have default values, for example the default settings for the motor on VI is “All Ports” for the port input terminal and “75” for the power setting. Context help clearly identifies these values if you mouse-over a VI.

Wire the commands together in the order you want them to execute. Every VI has terminals that the wires connect to.

Double-click on an empty spot on the block diagram to create a textbox to write a comment.

5. Have the students copy the program, then test their cars to determine the proper value of  $x$  that will allow them to drive their car 6”. Encourage the students to keep track of time versus distance in a table as shown below. Then, students can create a graph with this information to find the exact time needed to travel 6”.

Time (s)	Distance (in)



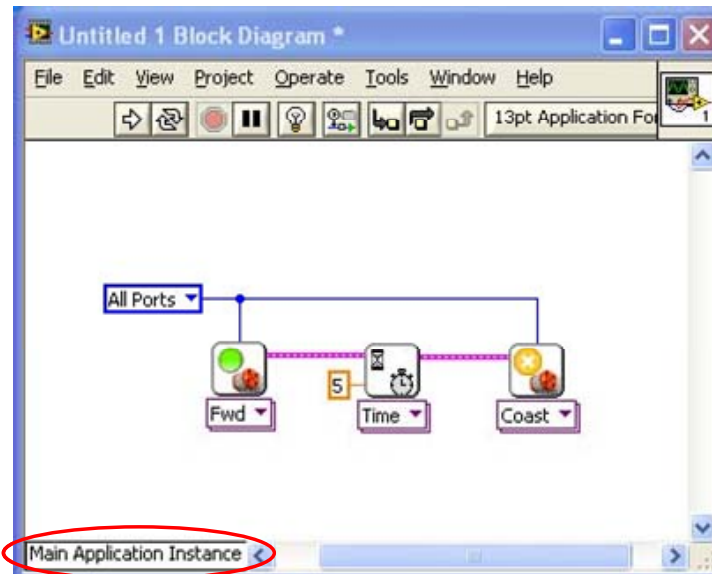
6. Instead of modifying the time constant, students can also modify the motor power level. The students can think about the tradeoff between how fast their rescue vehicle can deliver the supplies to the stranded mountain climbers versus the accuracy (driving the exact distance). Also consider the safety of the cargo, as stopping too abruptly may break fragile supplies.
7. The students should add comments to their code so that the next time they look at their code they know what the code is supposed to do. This also allows other people to easily understand their code, similar to the concept of drawing the design of their vehicle.
8. Once the students understand how to program the NXT in direct mode, explain the difference between direct mode and remote mode. They will need to use this mode to drive the rescue vehicle 6’ (the second challenge). Show them how to switch between the two modes using the File menu. When you are in direct mode, you see a white box that says “Main Application Instance” in the lower left-hand corner of the front panel and block diagram (see below). When you are in remote mode, you see an orange box that says the name of the NXT that is connected to the computer; in this case the name of the NXT attached is named Darby (see below).

## Lesson 1

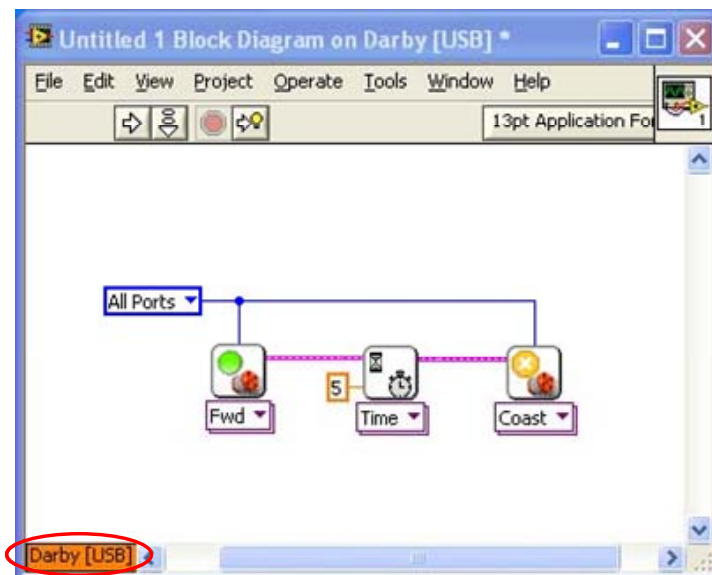
## Rescue Rover

### Programming Tip

If Control-T (PC) or Command-T (Mac) tiles the LabVIEW front panel and block diagram instead of switching between the direct and remote mode, go to Tools>NXT Preferences>click on the Miscellaneous tab in the NXT Module Systems Preferences dialog box that opens. Then check off the option that allows you to use Control-T (PC) or Command-T (Mac) shortcut.



Direct mode



Remote mode

## Lesson 1

## Rescue Rover

### Useful Circle Formulas

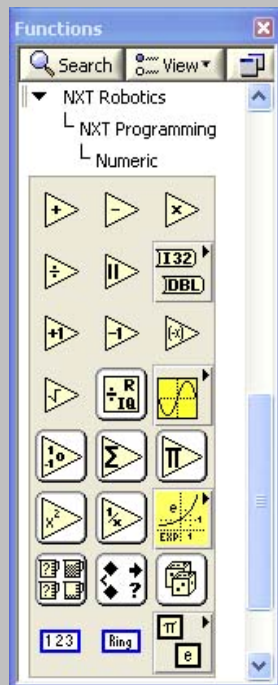
Circumference =  $2*\pi*\text{radius}$

Linear distance =  $\text{radius}*\text{angle (radians)}$

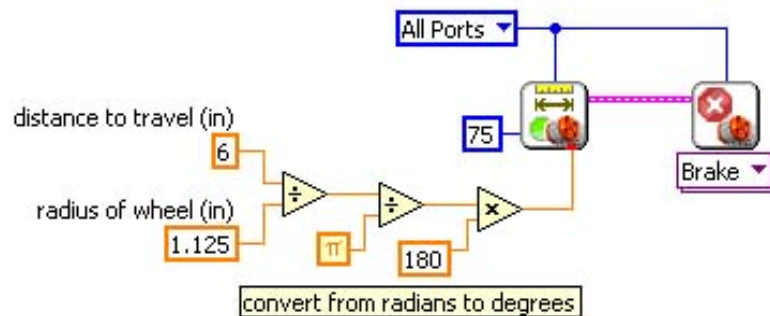
$180^\circ = \pi$  radians

### Programming Tip

Numeric palette:



9. Have the students test out their program in remote mode. Their rescue vehicle should still travel 6".
10. Ask the students to program their vehicle to drive for 6' using remote mode. They should be able to find the exact time needed using the same method as before.
11. Once the students successfully get their car to drive for 6' using the trial and error method, challenge them to mathematically calculate the number of degrees the wheels need to rotate to travel 6'. Instead of controlling the rescue vehicle with time, they can control it using the rotation sensors that are built into the motors. The drive distance VI, located in the Complete palette, takes distance (in degrees) as an input. To calculate the distance in degrees, the students will need to divide the linear distance to travel by the radius of the wheel and convert that output which is in radians into degrees. Remind the students to add comments their code.



12. Once again the students should consider the tradeoff between speed and accuracy (the students can adjust the power going into the motors, the default power level is 75).

**Part IV: Class Discussion / Reflection****20 minutes**

1. When all the students have completed the challenge, have the students present their design and demonstrate their final solution. Ask the students:
  - a. What did you brainstorm (car or container) that you did not use?
  - b. What difficulties did you overcome in building? In programming?
  - c. Were you able to get your car to stop at the exact location?
2. What is the trade-off between speed and accuracy?
3. Instead of using trial and error to find the how long the motor has to run to travel a certain distance, how can you calculate it? What is the relation between motor rotation, speed, distance, and time?
4. Go over the engineering design process with the students. Refer to things that happened during the activity that illustrate the different steps. Discuss with the students which steps they completed naturally and which ones they skipped and how the skipped steps might have been helpful.

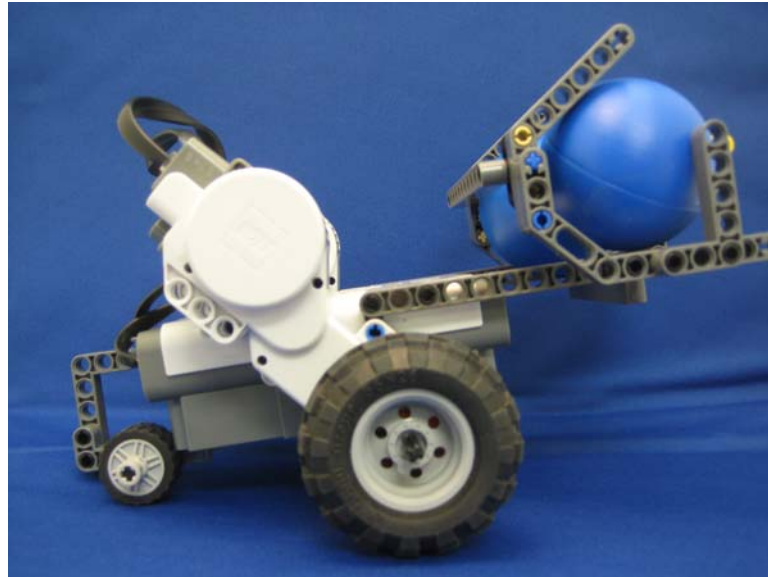
**Extensions**

---

1. Programming: Program the rescue vehicle to return to the start.
2. Building: Build a device that can pick up and drop off cargo, instead of simply carrying it. (Hint: this will require another NXT motor.)
3. Teamwork: Collect cargo directly from another group's rescue vehicle.

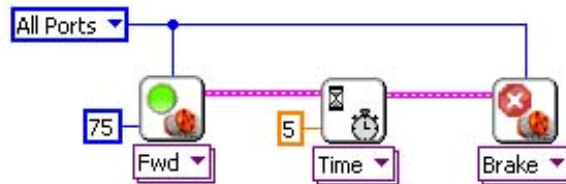
Sample Projects and Photos

Photo of sample NXT car with wires:

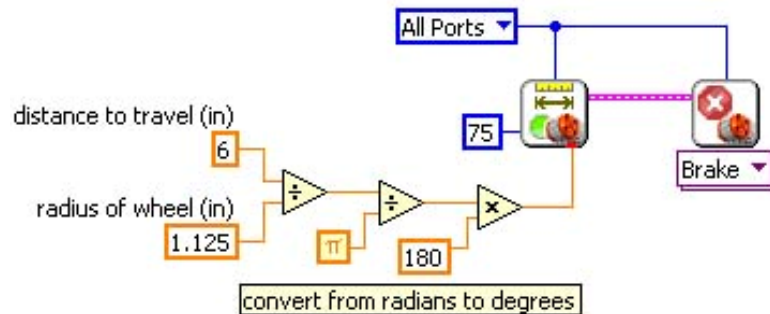


Sample Code:

Trial and error method:



Mathematically determined method:



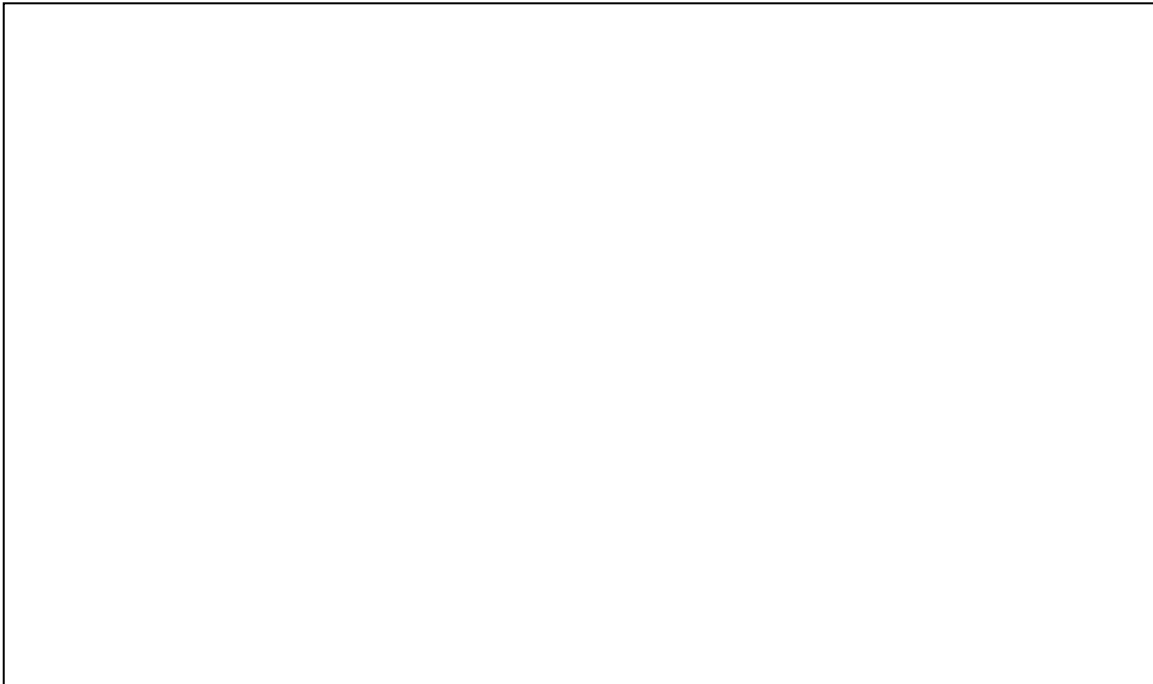
# Lesson 1 Engineer's Journal

## Rescue Rover

1. Write down and/or draw your rescue vehicle design ideas:



2. Create a sketch of the mechanism you constructed, label the parts, and give the overall dimension of your vehicle:



3. What features does your prototype have?

---

---

4. How do you plan to program your car to drive 6”?

---

---

---

5. How long did it take your car to drive 6”?

---

---

6. How long did it take your car to drive 6’?

---

---

7. What are some of the challenges that you encountered in this activity?

---

---

---

---

8. What did you learn from this activity?

---

---

---

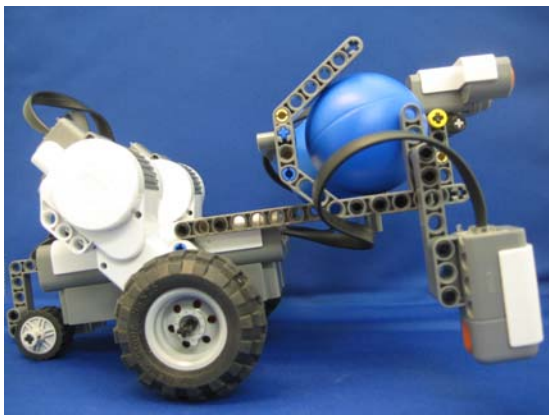
---

# Ledge Leader

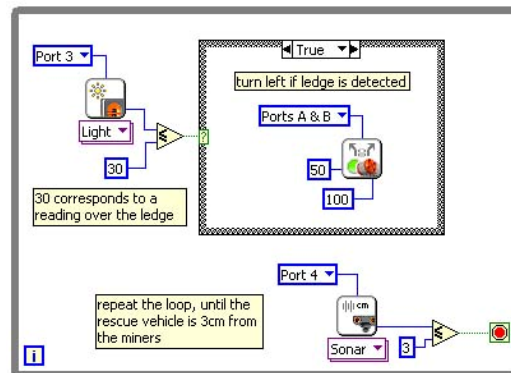
## Overview

In the last challenge, the students built prototypes of autonomous rescue vehicles to drive specified straight line distances to deliver supplies to the stranded mountain climbers. However, it is unlikely that there is a clear, straight line path up a mountain. In this lesson, the students will learn how to improve the intelligence of their rescue vehicle so that it can respond to the environment in real-time. To do this the students will add two sensors to their rescue vehicle (1) a sensor to detect the edge of the path and (2) a sensor to detect when it has reached the stranded mountaineers. The students will also learn how to write more advanced programs in LabVIEW Education Edition using case structures and while loops.

Sample Vehicle



Sample Code:



Expectations	Evidence
<p>Students should be able to:</p> <ul style="list-style-type: none"> <li>• Build sturdy sensor attachments on their rescue vehicle.</li> <li>• Use LabVIEW to program their rescue vehicle to stay on the path and stop when it reaches the target location.</li> <li>• Use the engineering design process to solve the given challenge.</li> </ul>	<p>Evidence of learning found in:</p> <ul style="list-style-type: none"> <li>• A NXT vehicle with two sensors securely attached.</li> <li>• Commented LabVIEW code.</li> <li>• Rescue vehicle that can complete the challenge.</li> <li>• Engineer's journal.</li> </ul>



# Lesson 2

# Ledge Leader

## Suggested Time

90 minutes

## Vocabulary

(See Appendix L)

- Sensor
- LED
- Case structure
- Boolean
- While loop

## Materials

### Each student:

- Engineer's Journal

### Each student group:

- LEGO MINDSTORMS kit
- Computer with LabVIEW Education Edition

## Teacher Preparation






- Build a sturdy car with an ultrasonic sensor attached as an example.
- Distribute Engineer's Journals.
- Make a course for the rescue vehicles to navigate. The ledge can be simulated using black electrical tape if the students are using light sensors or an elevated platform (e.g. tabletop) if the students are using ultrasonic sensors to detect the ledge.

## Challenge

Instead of navigating a straight path, this time the rescue vehicle has to drive up an unknown path to the stranded mountain climbers. The rescue vehicle has to stay on the path and not fall off the ledge on the right-hand side of the path. The rescue vehicle will need two sensors: (1) a sensor to detect the edge of the path and (2) a sensor to detect when it has reached the mountaineers.

## Background

Robots, like humans, are able to identify information about the environment to determine the best path to travel. Humans use their five senses (sight, hearing, taste, smell, and touch) to sense the environment and robots use **sensors** to accomplish the same task. The LEGO MINDSTORMS NXT kit comes with the following sensors:

<p><u>Light Sensor</u>: detects light and color</p> 	<p><u>Touch Sensor</u>: detects when the button is released, pressed, or bumped</p> 
<p><u>Ultrasonic Sensor</u>: measures distance in centimeters and inches</p> 	<p><u>Sound Sensor</u>: measures the volume of sound in decibels</p> 
<p><u>Rotation Sensor</u>: measures the number of rotations in degrees</p> 	

## Lesson 2

## Ledge Leader

The sensors (except the built-in rotation sensor in the motor) connect to port 1, 2, 3, or 4 of the NXT brick and pass information along the NXT wires, through the ports to the NXT “brain”.

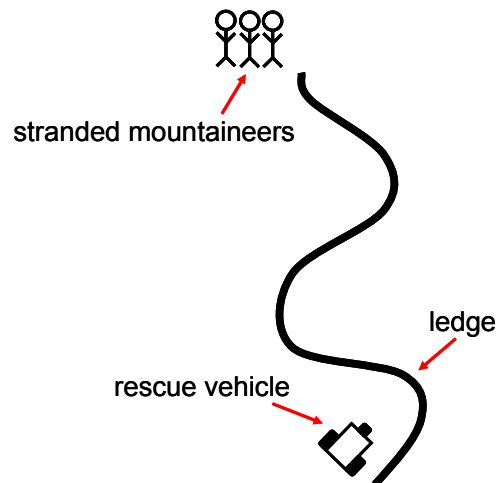
### Instructions

---

#### Part I: Introduction

**15 minutes**

1. Review the challenge from last time and ask the students to remind you what the design constraints were for that challenge.
2. Introduce the new challenge to the students. They will be revisiting the mountain rescue scenario, but in a more realistic setting. Instead of a clear, straight path, the rescue vehicle has to navigate up an unknown perilous route to the stranded mountaineers. There is a ledge on the right-hand side of the trail and the rescue vehicle has to stay on the path without falling off. The rescue vehicle also has to stop when it reaches the mountaineers.



### Helpful Hint

---

You can use the NXT Remote Control application, located under Tools/NXT Applications, to show your students the sensor values displayed on the screen of the NXT brick.

For more information on the NXT Remote Control see the “NXT Remote Application” help topic under NXT Module Basics/LabVIEW NXT Module Tour/NXT Module Menu Items/Applications, in the LabVIEW NXT Module Help (under the Help file menu).

The goal of this activity is to help the students fine-tune their prototype in the indoor environment (the classroom) before building the actual rescue vehicle with the TETRIX construction set and taking the challenge to the outdoor environment.

3. Robots, like humans, can detect their immediate environment through their senses known as sensors. The students’ robots will need sensors to accomplish the challenge in this lesson. Go over the sensors that are included with the NXT kit and demonstrate how each sensor works. Also show the students how they can view sensor values on the screen of the NXT brick (see Appendix D).

## Lesson 2

## Ledge Leader

### Helpful Hint

When the light source, **light emitting diode (LED)** is off, the sensor can be used to detect ambient light in the environment.

When the LED is on, the sensor can be used to measure reflected light. Use this mode when the light sensor is close to the surface or when the environment is dark.

### Real World Connection

Bats are blind, but use ultrasound to detect objects and fly through the air without running into objects (echolocation). The ultrasonic sensor uses this same technology to detect objects.

### Helpful Hint

The light sensor or ultrasonic sensor can be used to detect the edge of the path and the ultrasonic sensor or the touch sensor can be used to detect the mountaineers.

Demonstrating the NXT sensors:

Light sensor	Compare light sensor values for: <ul style="list-style-type: none"><li>• light colored objects versus dark color objects</li><li>• bright room versus dark room</li></ul>
Touch sensor	Explain the difference between pressed and bumped.
Ultrasonic sensor	Compare ultrasonic sensor values for objects (e.g. your hand) at different distances from the ultrasonic sensor.
Sound sensor	Compare sound sensor values of a loud sound versus a quiet sound.
Rotation sensor	Turn the motor (in both directions) to show how the direction of rotation affects the rotation sensor value.

4. As a class identify the two tasks the rescue vehicle has to accomplish (step 1 of the engineering design process).
5. Ask the students to develop possible solution(s) for the problem (step 3 of the engineering design process). They should experiment with the sensors to get a better idea of how the sensors work. There is more than one possible solution to the problem. Give the students a few minutes to brainstorm which sensors can be used for each task and why. Students can either do this task individually or in pairs. They should write down their ideas in their Engineer's Journal.
6. As a class, go over all the possible solutions and ask each group to choose what they think is the best possible solution (step 4 of the engineering process). This will be the solution they will implement.

## Lesson 2

## Ledge Leader

### Hardware Tip

The sensor used to detect the edge of the path should be placed close to the ground and attached to the side of the vehicle in such a way that the robot can detect the edge and respond to it before the robot falls off.

Remember to plug the sensors into ports 1, 2, 3, or 4 of the NXT brick using the provided connector cables so that the sensor values can be sent to the NXT "brain".

### Part II: Building

**15 minutes**

1. Return the rescue vehicles (from lesson 1) to each group.
2. Give the students a few minutes to brainstorm where and how the selected sensors should be attached to their rescue vehicles.
3. Have the students design and build the sensor attachments for their rescue vehicles. The rescue vehicle has to meet the following design constraints from last time:
  - a. Smaller than 12"x 9"x9".
  - b. 2 motors.
  - c. A container to carry supplies.
  - d. Sensor to detect edge of path.
  - e. Sensor to detect when it has arrived at the mountaineers.
  - f. Sturdy.
4. The rescue vehicle has to pass the following tests to be deemed sturdy:
  - a. Shake test –The teacher must be able to shake the rescue vehicle without pieces falling off.
  - b. Drop test – The teacher must be able to drop the rescue vehicle from the height of his/her ankle without the car breaking.
  - c. Steady test – The sensors must remain securely attached to the rescue vehicle as it drives around.
5. Rescue vehicles that are not sturdy need to be redesigned.
6. As the students finish building their sensor attachments, remind them to document their design by drawing diagrams of their finished designs in their Engineer's Journal.
7. Have the students set the car aside as you introduce programming portion of the activity.

## Lesson 2

## Ledge Leader

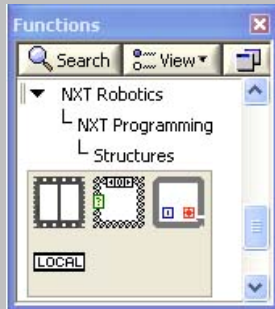
### Hardware Tip

The precision of the ultrasonic sensor is +/- 3cm. Also the ultrasonic sensor does not work well for distances less than 5 cm.

### Programming Tip

Flowcharts help students think through the programming logic. Once the students understand the flow of the program they need to write, they can easily translate their flowchart into code.

Both the case structure and while loop can be found on the Structures palette:

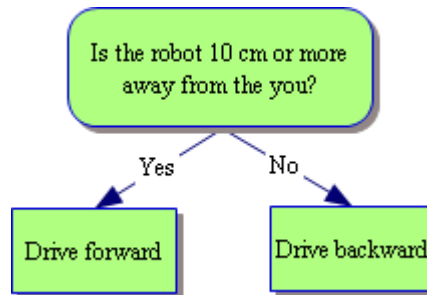


For more information on case structures see the "Making a Decision (If Statements)" help topic under Programming Structures in the LabVIEW NXT Module Help (under the Help file menu).

### Part III: Programming

40 minutes

1. The students will learn about two programming structures, **case structures** and **while loops** that will allow them to create more sophisticated programs. Use an example to demonstrate the functionality of these two structures. For instance, program an NXT car to follow 10 cm behind you. When you are greater than 10 cm away, the NXT car drives forward. When you are less than 10 cm away, the NXT car backs up.
2. Diagram the program you will write in a flowchart, like the one shown below. Ask the students what the robot needs to do when you are greater than 10 cm away? What about when you are less than 10 cm?



3. Show the students how to write this program in LabVIEW using the case structure. Case structures allow you to choose between different behaviors based on the current condition. The selector terminal, denoted by the green question mark on the frame of the case structure, determines which case executes (see below). In this example, the case selector is a **Boolean**. A Boolean is a logical data type that is either true or false.

## Lesson 2

## Ledge Leader

### Helpful Tip

The orientation of your motors depends on the construction of your NXT car. You may have to switch forwards and backwards in your program to match the desired response of the NXT car.

### Programming Tip

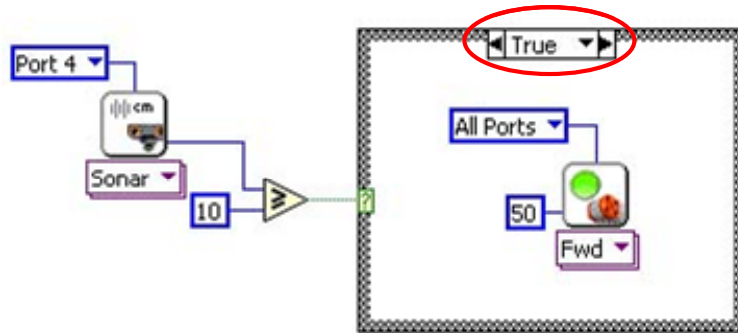
Note that both the true and false case is displayed to show the code for each case. To view the different cases in your LabVIEW code, click on either the down arrow or the left/right arrows located at the top of the frame of the case structure.

When highlight execution is turned on, the light bulb is yellow. Highlight execution only works in direct mode.

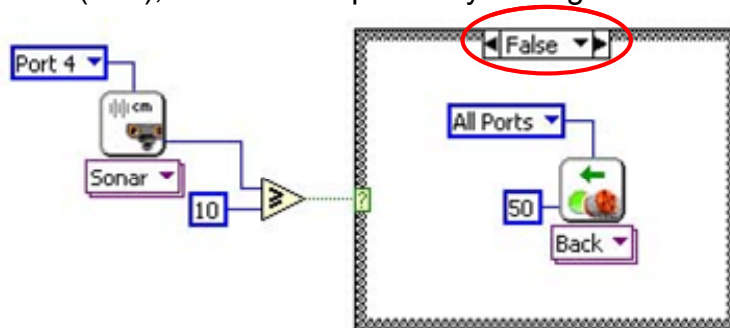
### Programming Practice

First write your code in direct mode so that you can use the LabVIEW debugging tools, like highlight execution, to help you debug the program. Then run your program in remote mode so that you can run your robot untethered from the computer.

For information on how to debug a LabVIEW program see the "Debugging and Troubleshooting" section in LabVIEW NXT Module Help (under the Help file menu).

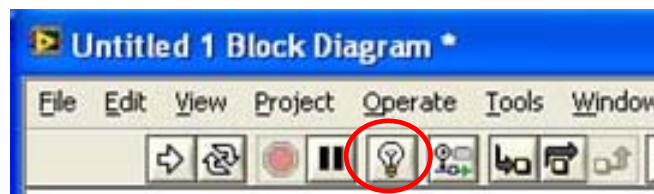


If the ultrasonic sensor measures a value greater than or equal to 10 cm (true), the robot responds by driving forward.



If the ultrasonic sensor measures a value less than 10 cm (false), the robot responds by driving backwards.

4. Run the example code. Ask the students why the car runs only in one direction before it stops? The reason is that the code executes once and the program is finished. Show the students this by running the program in direct mode with the highlight execution turned on. The highlight execution tool is located on the block diagram toolbar (see below).



Highlight execution is a useful **debugging** tool that slows down the execution of your program and allows you to watch the movement of data through your code. You should see the value read by the ultrasonic sensor, the result of the comparison, and the selected case.

## Lesson 2

## Ledge Leader

### Programming Tip

An end condition has to be wired to the stop sign for the code to not be broken.

Loop counters in LabVIEW start at 0, not 1.

For more information on while loops see the "Using Loops" help topic under Programming Structures in the LabVIEW NXT Module Help (under the Help file menu).

### Hardware Tip

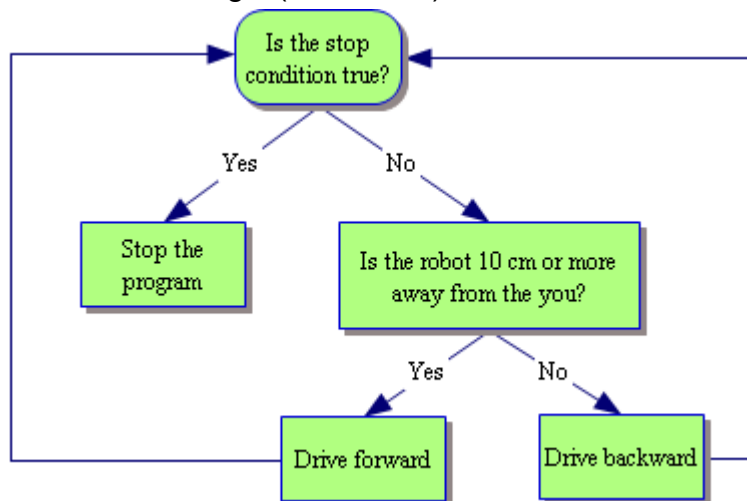
When you are running the code in direct mode, you can't stop the program by pressing the dark gray button on the NXT brick because the NXT Shell will restart. For more information about the NXT Shell see "NXT Shell" help topic under Downloading and Running Programs in the LabVIEW NXT Module Help (under the Help file menu).

- Now introduce the while loop. While loops repeat code inside the loop until the end condition of the loop is met. LabVIEW while loops have 2 elements: (1) the loop counter denoted by the letter *i* and (2) an end condition denoted by the stop sign.

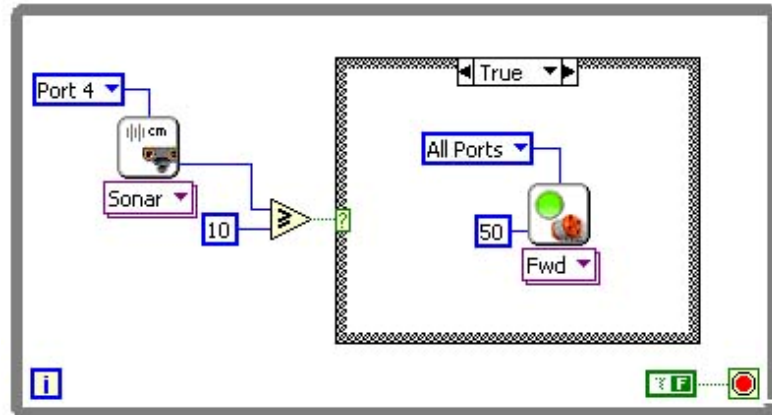


In general, while loops are used for situations where you want the code to repeat for an undetermined number of times. In this activity the students will not be using the loop counter, just the end condition.

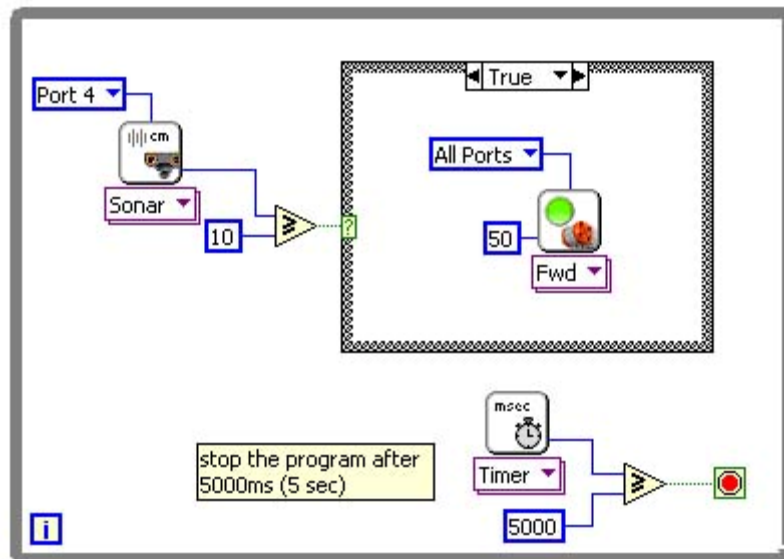
- Add the infinite while loop logic to the flow diagram to demonstrate the logic (see below).



- Show the students how to add the infinite loop to the LabVIEW code. Encase the case structure code in a while loop and wire a false constant to the end condition by right-clicking on the end condition stop sign and then create a constant (see below). This will make the code run forever because the stop condition is always false. To stop the program on the NXT brick, you will need to click the stop button on the block diagram tool bar.



8. Next show the students how to create a finite while loop—a program that ends when the stop condition is met. Instead of wiring in a false constant to the stop condition, wire in a function that outputs a Boolean, like a comparison as shown in the code below.



The students will want to use finite while loops and case structures for this activity.

9. Have the students replicate the example program and test it, so they can gain a better understanding of how case structures and while loops operate.
10. Give the students a few minutes to draw a flowchart of their program in their Engineer's Journal. The students have already selected the sensors they will use to detect the mountain edge and the stranded mountaineers.



### Helpful Hint

If the students used the light sensor to detect the ledge, the threshold level for the ledge will vary depending on the lighting situation in the room.

11. Once the students finish outlining their code, ask them to implement the code in LabVIEW. Set up a course where the students can test out their code. The students should first write their code in direct mode and use the debugging tools within LabVIEW to debug their code. The students should debug their code until their rescue vehicle can consistently complete the challenge. Ultimately the students should test their robots in remote mode. Remind the students to add comments to their code. (Reference sample code at the end of the lesson.)

### Part IV: Class Discussion / Reflection

**20 minutes**

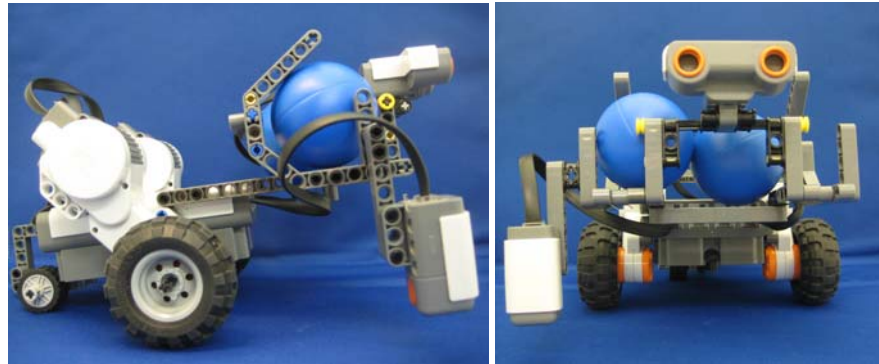
1. When all the students have completed the challenge, have the students present their design and demonstrate their final solution. Ask the students:
  - a. How does the rescue vehicle detect the ledge? The mountaineers? What does it do in each case?
  - b. What difficulties did you overcome in building? In programming?
2. Discuss the need for calibration, as sensor data are sensitive to environmental conditions. Demonstrate the difference by having the students run their robots in the dark versus having the classroom lights on.
3. Review the engineering design process and ask the students to identify which steps they completed and which ones they skipped. How might the skipped steps have been helpful?

### Extensions

1. Programming: Program the rescue vehicle to return to the starting point.
2. Building: Construct a flag raising mechanism that will signal when the vehicle arrives at its destination.
3. Teamwork: Program the robot to speed up the supply delivery by using 2 robots in the same constrained environment.

## Sample Projects and Photos

Photo of sample NXT car:

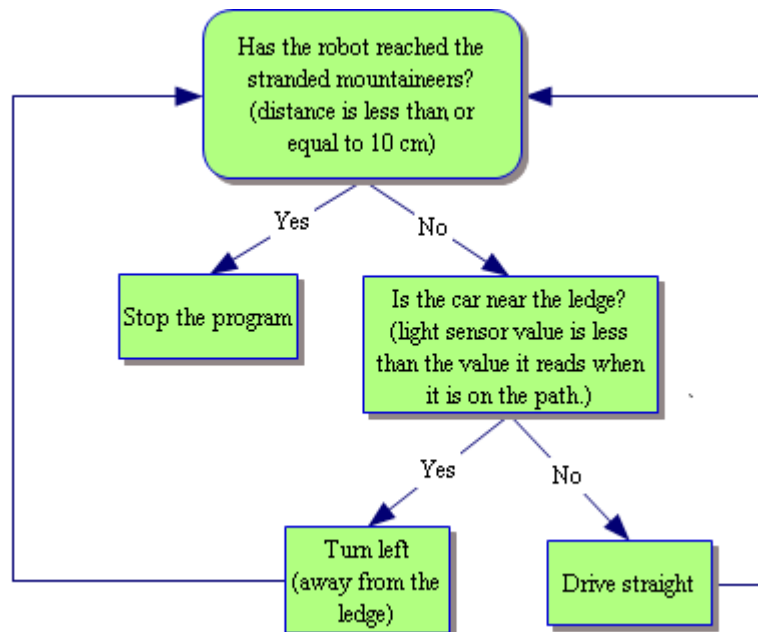


Side View

Front View

This rescue vehicle uses the light sensor to detect the ledge and the ultrasound sensor to detect the mountaineers.

Sample Program Flowchart:



## Lesson 2

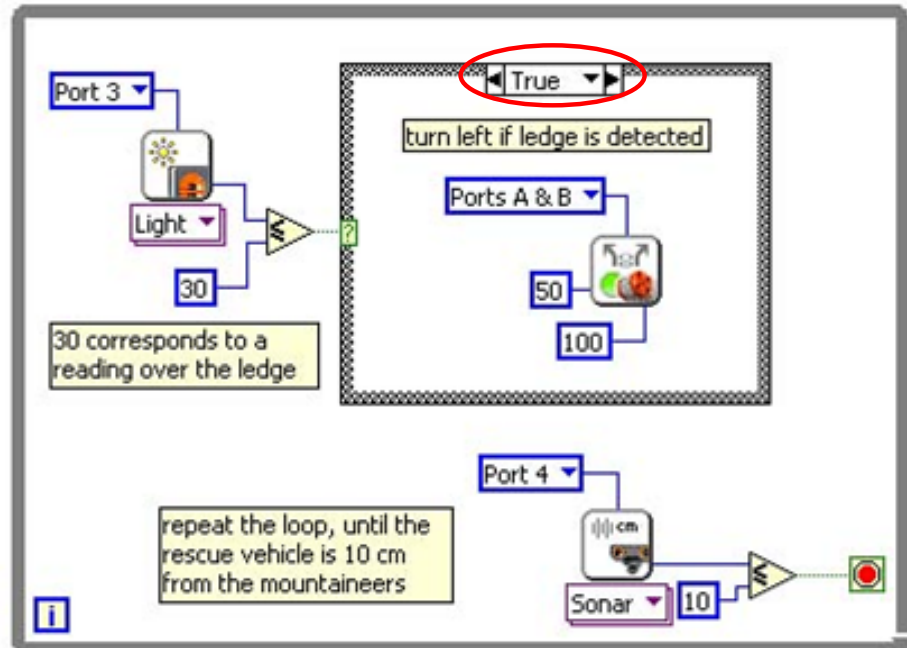
## Ledge Leader

### Programming Tip

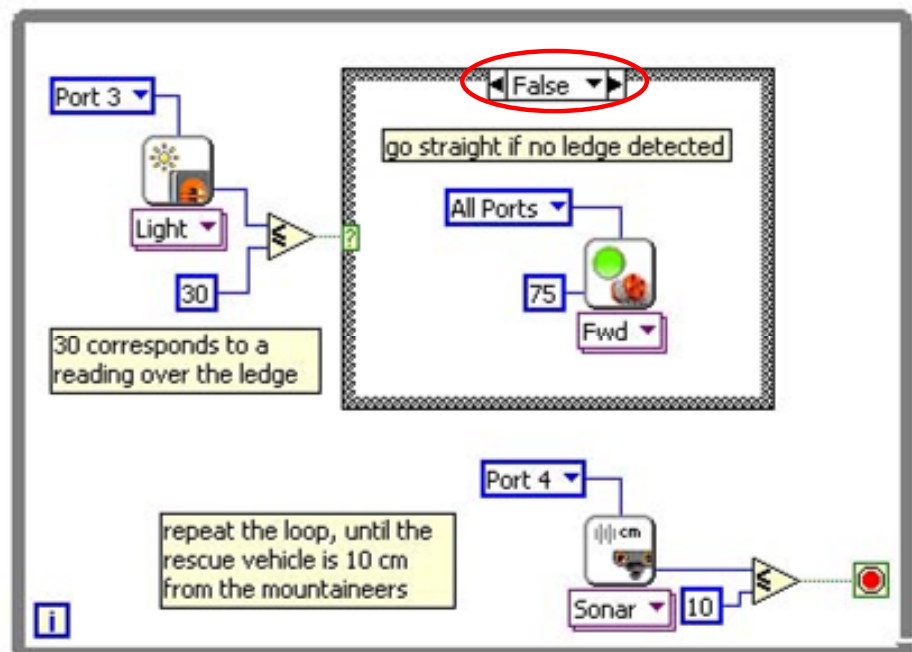
Modify the steering input value on the Steering On VI to control the sharpness and direction of the steering.

Sample Code:

For the True case, the vehicle turns left (motors turn in opposite directions).



For the False case, the vehicle drives straight (both motors on).



## Lesson 2 Engineer's Journal

### Ledge Leader

1. What sensors can you use to detect the ledge?

---

---

---

2. What sensors can you use to detect when the rescue vehicle has reached the stranded mountain climbers?

---

---

---

3. Which sensors will you use and why?

---

---

---

4. Create a sketch of the mechanism you constructed. Remember to label the parts.



5. How does the rescue vehicle detect the ledge? What does it do when it detects the edge? What does it do when it does not detect the edge?

---

---

---

6. How does the rescue vehicle detect when it has reached the stranded mountain climbers?

---

---

7. Draw a flowchart of the program.



8. What are some of the challenges that you encountered in this activity?

---

---

---

---

9. What did you learn from this activity?

---

---

---

---

10. What steps of the engineering design process did you use?

---

---

---

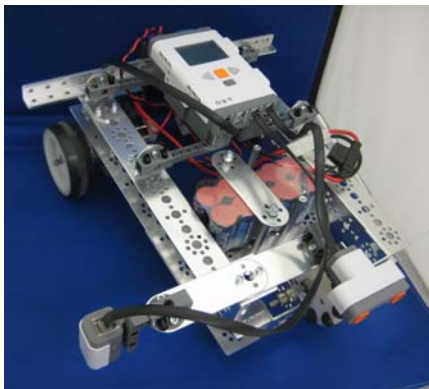
---

# Candid Climber

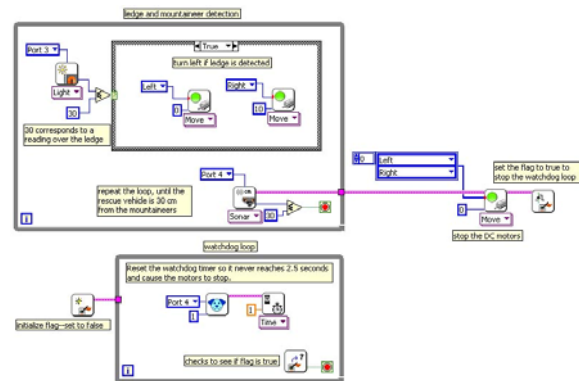
## Overview

In Lessons 1-2, the students used the NXT kit to prototype rescue vehicles to traverse a flat path to rescue stranded mountain climbers. In this lesson the students will turn the prototype into the “real thing” and take the rescue mission outdoors, using the TETRIX™ building materials to climb up a hill. The rescue vehicle has to stay on the path and not fall off the ledge, as well as stop when it reaches the climbers. In this challenge the students will learn TETRIX construction and programming techniques.

Sample Vehicle



Sample Code



Expectations	Evidence
<p>Students should be able to:</p> <ul style="list-style-type: none"> <li>• Build a sturdy 2 motor rescue vehicle using the DC motors.</li> <li>• Build sturdy sensor attachments on their rescue vehicle.</li> <li>• Use LabVIEW to program their rescue vehicle to stay on the path and stop when it reaches the target location.</li> <li>• Use the engineering design process to solve the given challenge.</li> </ul>	<p>Evidence of learning found in:</p> <ul style="list-style-type: none"> <li>• A TETRIX vehicle with 2 DC motors and 2 sensors securely attached.</li> <li>• Commented LabVIEW code.</li> <li>• Rescue vehicle that can complete the challenge.</li> <li>• Engineer's journal.</li> </ul>

## Lesson 3

## Candid Climber

### Suggested Time

180 minutes

### Vocabulary

(See Appendix L)

- DC motor
- Torque
- Chassis
- Watchdog

### Materials

#### Each student:

- Engineer's Journal

#### Each student group (4):

- LEGO MINDSTORMS kit
- TETRIX kit
- Computer with LabVIEW Education Edition

### Teacher Preparation

- Provide students with pictures of sample TETRIX vehicles.
- Distribute Engineer's Journals.
- Make an outdoor course for the rescue vehicles to navigate.

### Challenge

Build a TETRIX rescue vehicle that can climb up a steep outdoor hill (at least 20 degree incline) and navigate a sinuous path to reach the stranded mountaineers. The rescue vehicle has to stay on the path and not fall off the ledge on the right-hand side of the path. In addition to 2 **DC motors**, the rescue vehicle will need two sensors: (1) a sensor to detect the edge of the path and (2) a sensor to detect when it has reached the mountaineers.

In order to have enough parts leftover to build a robotic arm to deliver supplies in Lesson 4, the students will be limited to half of the parts or less in the TETRIX kit to build their vehicles.

### Background

Prototyping is a common practice to flesh out a solution. However, when it comes to implementing the solution in the field, more durable materials are needed. To solve the mountain rescue challenge in a real-world environment the students will utilize the TETRIX metal building system from Pitsco, along with the MINDSTORMS NXT intelligence from LEGO (the NXT brick "brains" and NXT sensors).

This challenge involves testing the robot in an outdoor environment. You will need to select a hill or inclined path of no more than 45 degrees for the students' robots to navigate. The HiTechnic motors in the TETRIX kit are powerful and the limiting factor with the robot's ability to climb up a hill will be the friction between the wheel and the ground. If the motor is providing enough **torque**, the wheel will rotate with slipping. If there motor is not providing enough torque, the wheels will not rotate. To increase the amount of torque, the students can gear up their motor to increase the gear ratio. For more information on gear ratios, see Appendix E.

Depending on time constraints and your students' robotics skill level you may want to specify which sensor the students should utilize to detect the edge of the path. The ideal sensor for the ledge detection depends on the terrain.



## Lesson 3

## Candid Climber

### Discussion

A prototype is a model of a design on which tests can be performed to evaluate whether the design should be used for the final product.

The LEGO MINDSTORMS technic pieces are useful prototyping tools because the pieces are modular and snap together easily, allowing engineers to quickly test out their designs. However, the plastic LEGO MINDSTORMS technic pieces are not ideal for the harsh outdoor environment.

### Helpful Hint

To give the students the “big” picture, explain that they will build the robot chassis in this lesson, a robotic arm in Lesson 4, and assemble the two parts in Lesson 5 to create the final rescue robot.

### Classroom Management

This activity works best if the students work in teams of 4 per TETRIX kit.

### Hardware Tip

Needle nose pliers are useful for tightening screws in hard to reach places.

### Instructions

#### Part I: Introduction

**20 minutes**

1. Prepare the students for the final design phase by asking them to reflect on the work they have done thus far. A discussion could involve the following questions:
  - What is the purpose of prototyping?
  - What are the prototyping strengths of the LEGO MINDSTORMS kit?
  - How well do you think the prototype rescue vehicle will hold up in the real environment? Would the LEGO MINDSTORMS construction material be appropriate for the actual rescue vehicle?
2. Introduce the new challenge to the students. Tell the students that in this lesson they will begin building their final rescue robot using the TETRIX kit. They will focus on building the rescue vehicle **chassis**.

#### Part II: Building

**70 minutes**

1. Introduce the students to the TETRIX construction set. See Appendix F for a list of TETRIX parts.
2. To give the students practice building with the new material, ask them to build a sturdy square using 4 bar linkages and some screws and nuts (see below). A sturdy square is one that will maintain 90 right angles when you push on any 2 sides of the square. This activity is intended to help the students discover how to build sturdy TETRIX structures.



Front view



Back view

## Lesson 3

## Candid Climber

### Hardware Tips

The students will need a small flathead screwdriver to connect wires from the motor to the terminals on the motor controller.

Keep the wires clear of rotating components, like gears and motors.

Keep your fingers clear of the gears to avoid injury. The motors in the TETRIX kit are much stronger than the NXT motors and can cause a serious injury.

3. Depending on how comfortable you and/or your students are with the TETRIX construction set. You can either give your students building instructions for the sample vehicle (see Appendix G) or let your students design their own rescue vehicles. Remember to limit the students to half of the parts in the TETRIX kit. If the students are designing their own rescue vehicles, have the students make a detailed drawing of their rescue vehicle in their Engineer's Journal before they dive into the construction. They should think about where to put the NXT brick, sensors, DC motors, motor controller, and battery pack. The drawing will serve as a design plan and the students can modify their design as they build it. You should approve the drawings before they start building.
4. Share the following TETRIX construction resources with your students:
  - Creator's Guide (comes with the TETRIX kit)
  - Construction Tips:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/construction\\_tips.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/construction_tips.pdf)
  - Structures:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/structure.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/structure.pdf)
  - Motors, Gears, and Wheels:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/motors\\_gears\\_wheels.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/motors_gears_wheels.pdf)
  - Using LEGO with TETRIX:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/lego\\_tetrix.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/lego_tetrix.pdf)
  - Setting up DC Motors (tutorial video):  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/labview/testbed/tetrix\\_testbed.html](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/labview/testbed/tetrix_testbed.html)
5. Check the students' rescue vehicles and make sure that all the components are securely fastened.
6. As the students finish up building, tell them to draw their final design in their Engineer's Journal. They should label their drawing.

## Lesson 3

## Candid Climber

### Programming Tip



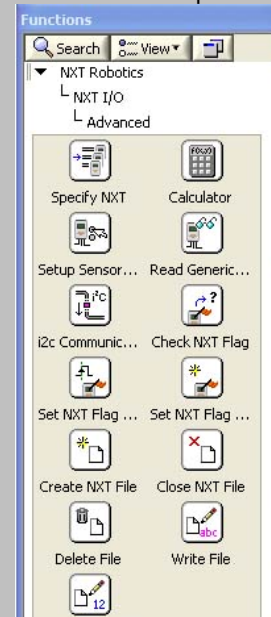
The watchdog timer is a safety feature that stops the motors after 2.5 seconds. To keep the motors on for more than 2.5 seconds the watchdog timer needs to be “fed” or reset before the timer reaches 2.5 seconds. See Appendix H for more details.

### Troubleshooting Tip

If the student’s robot is driving in circles instead of straight, he/she didn’t check reverse for one of the motors in the motor configurator. Have the student correct the problem in the motor configurator and update the program.

### Programming Tip

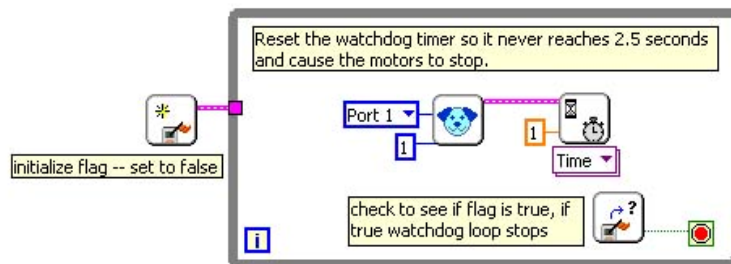
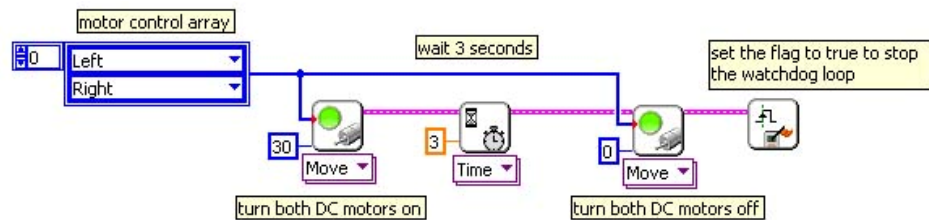
NXT flag VIs are located on the Advanced palette.



- After the students are done with the building portion of this activity, have them share:
  - Their design with other teams.
  - Any building challenges they encountered and how they address them.

### Part III: Programming, Testing, Redesigning 70 minutes

- Show the students how to program the NXT for use with TETRIX. First explain how to use the motor configurator to control the TETRIX DC motors and then explain the **watchdog** loop. For directions on how to set up the motor configurator for the DC motors see Appendix H. You can give your students a copy of the directions as a reference. For an in-depth explanation of the watchdog loop see Appendix H.
- As a first challenge, ask the students to write a program that will turn the DC motors on and drive the robot forward for 3 seconds (see below).



Some tips for programming the above code:

- Expand the control array to specify more than one motor
- Set power to 0 to stop the HiTechnic DC motors
- Use NXT flags to control the watchdog loop:
  - Initialize the flag to false
  - Watchdog loop continues until the flag is set to true

## Lesson 3

## Candid Climber

### Programming Tip

The students should be able to modify the rescue code they created in Lesson 2 for this challenge.

### Real World Connection

Repeatability is important in engineering because products need to have a certain usable lifetime. For example, the rescue robot may need to make multiple trips to deliver supplies to the stranded mountaineers. The robot will need to be perform reliably every time.

### Hardware Tip

When the students are not using the robot, they should turn the power off on the NXT brick and on the battery pack to avoid draining the batteries.

3. Once the students master the motor configurator and watchdog loop, they can go ahead and program their robot to complete the rescue mission:
  - Drive up the hill and staying on the path (don't fall off the ledge).
  - Stop when it reaches the stranded mountaineers.
4. Have the students test their programs and debug the code until they have a robot that can repeatedly accomplish the mission.
5. The students should add comments to the code to document their work.

### **Part IV: Class Discussion / Reflection** **20 minutes**

1. When all the students have completed the challenge, have the students present their design and demonstrate their final solution. Ask the students:
  - c. What programming challenges did they face?
  - d. What aspect of the challenge gave them the most trouble?
  - e. Did they have to redesign their rescue vehicle? If so, how did they redesign it?
2. Review the engineering design process and ask the students to identify which steps they completed and which ones they skipped. How might the skipped steps have been helpful?

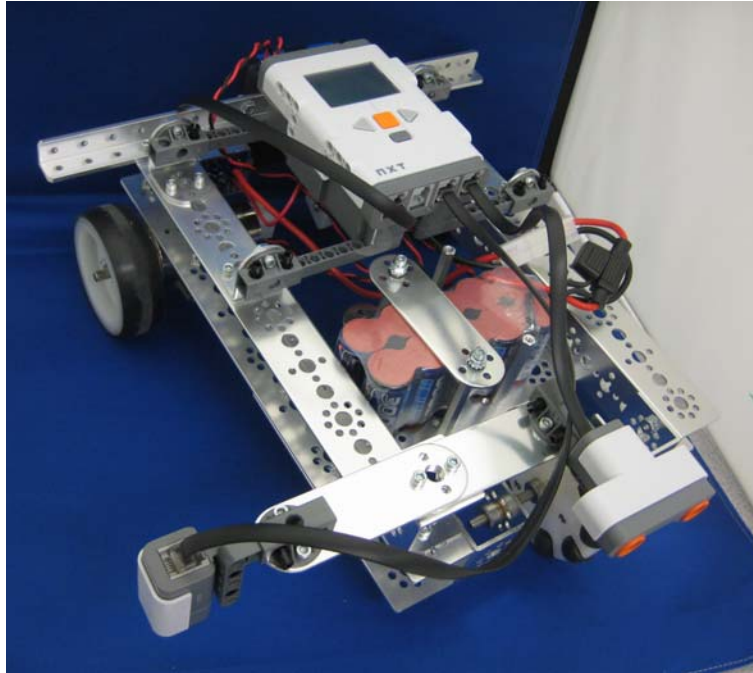
### **Extensions**

1. Programming: Program the TETRIX robot to navigate around obstacles.
2. Building: Optimize the design of the TETRIX robot by building a robot with as few parts as possible.
3. Teamwork: Program your robot to follow a foot behind another team's robot.

## Sample Project and Photos

---

Photo of sample TETRIX rescue vehicle:



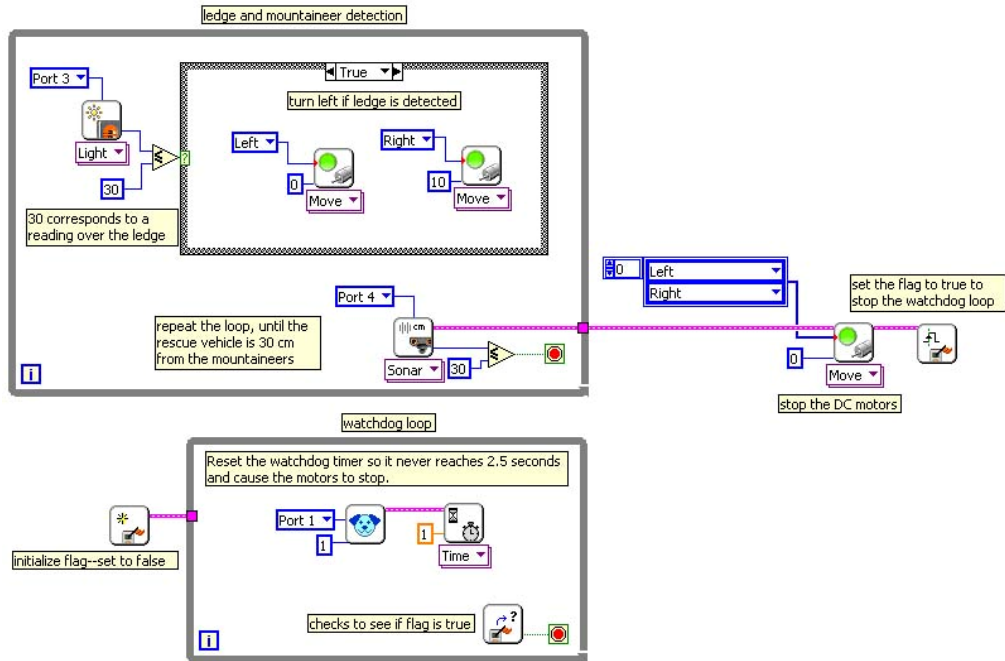
Port 1: HiTechnic motor controller

Port 3: light sensor

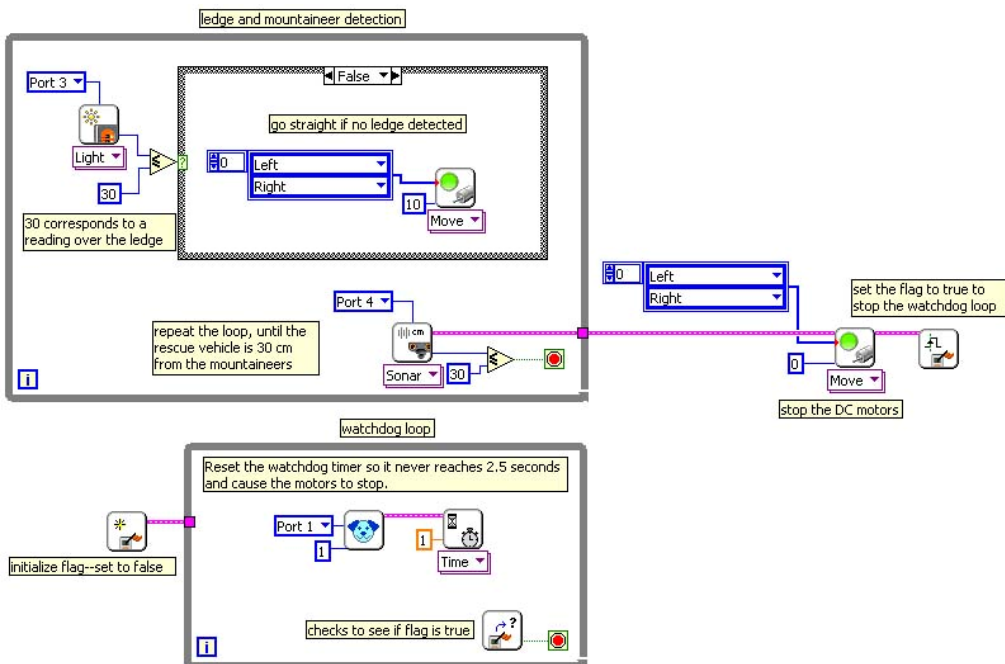
Port 4: ultrasonic sensor

Sample Project and Photos

Sample code:



True case: turn left (left motor off, right motor on)



False case: drive straight (both motors on)

## Lesson 3 Engineer's Journal

### Candid Climber

1. What is the purpose of a prototype?

---

---

2. What are the prototyping strengths of the LEGO MINDSTORMS kit?

---

---

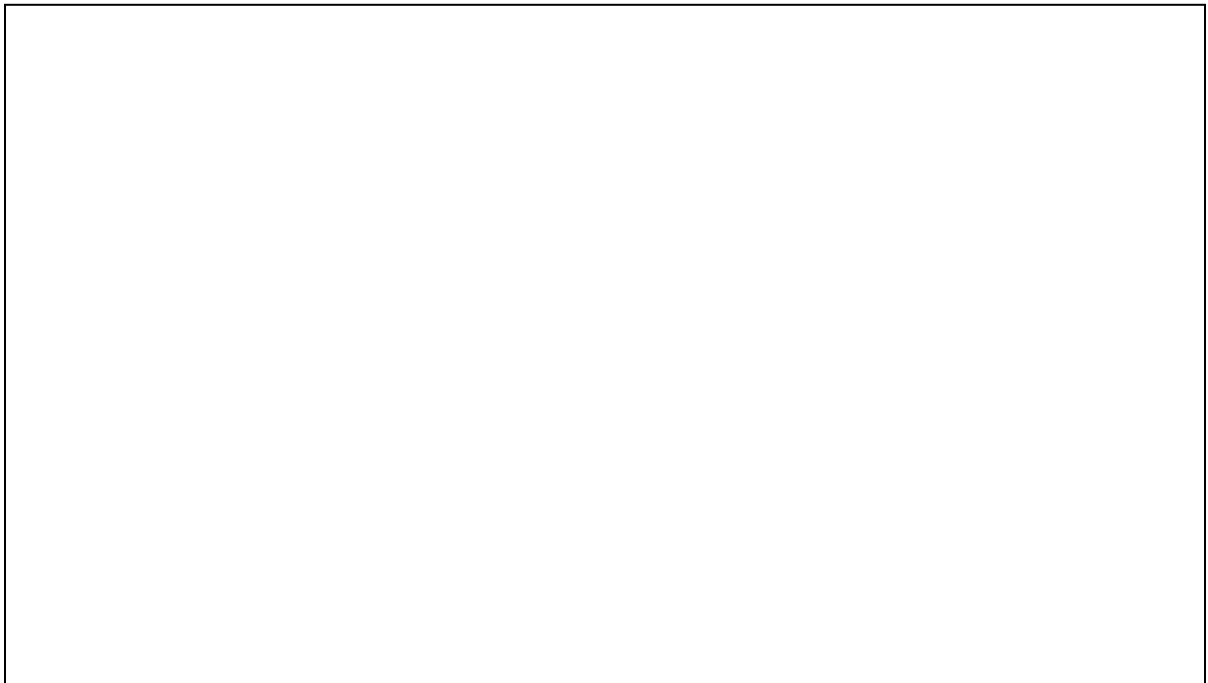
3. How well do you think the prototype rescue vehicle will hold up in the real environment? Would the LEGO MINDSTORMS construction material be appropriate for the actual rescue vehicle?

---

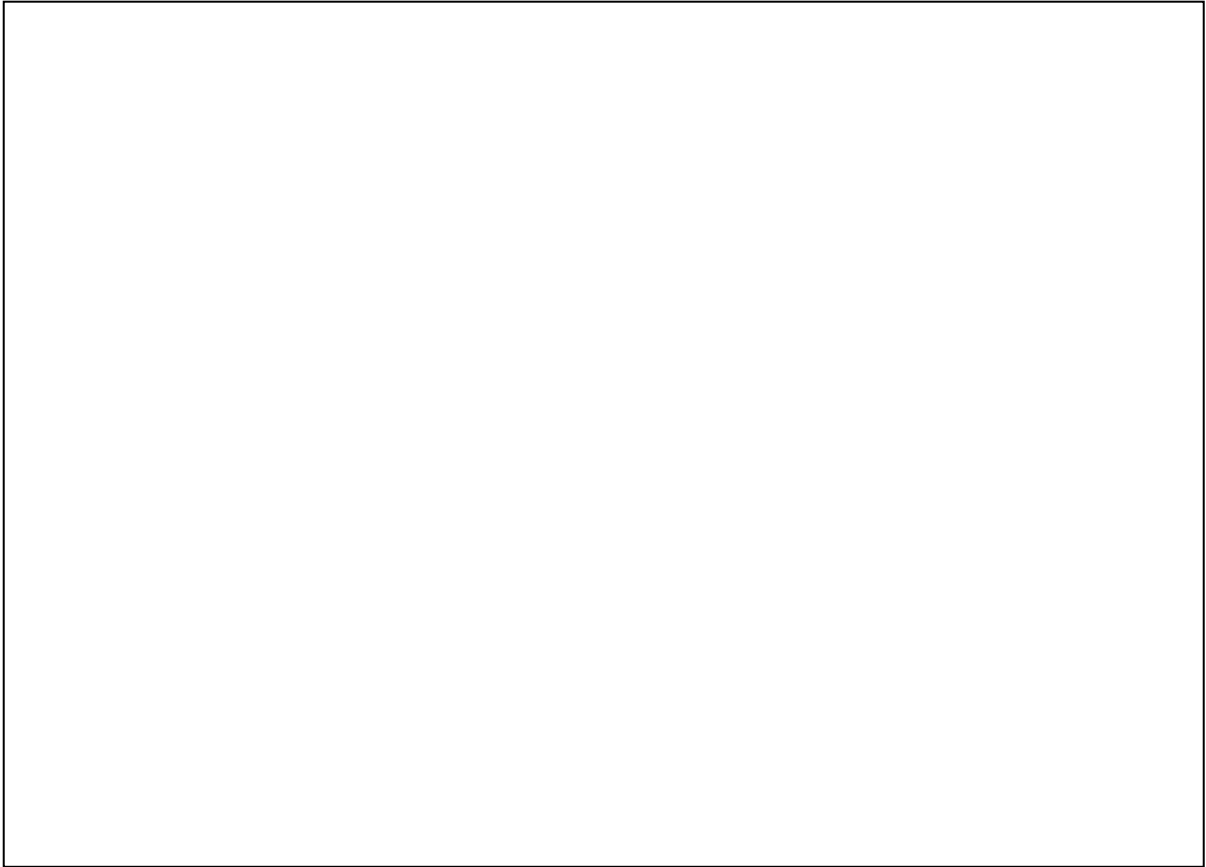
---

---

4. Make a detailed drawing of the TETRIX vehicle you plan to build. Think about where to put the NXT brick, sensors, DC motors, motor controller, and battery pack. Label your drawing.



5. Sketch your final robot configuration and label the components. Include multiple views if possible.



6. What are some of the challenges that you encountered in this activity?

---

---

---

---

7. What did you learn from this activity?

---

---

---

---



8. What steps of the engineering design process did you use?

---

---

---

---

# Reliable Reach

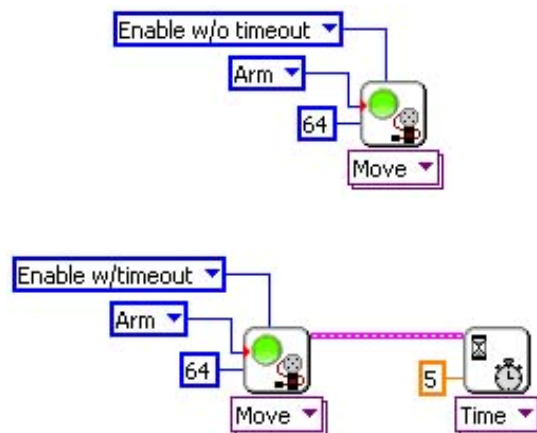
## Overview

Robots are used not only to transport things across the ground, but also as automatic lifting devices. In the mountain rescue scenario, the mountaineers are stuck on a small ledge that the rescue vehicle can not drive to. The rescue vehicle will need to have an arm to assist with the delivery of supplies. In this lesson the students will design, build, and program a TETRIX robotic arm. In addition to learning about servo motors and different robotic arm designs, the students will get the opportunity to explore the concept of torque.

Sample Robotic Arm



Sample Code



Expectations	Evidence
<p>Students should be able to:</p> <ul style="list-style-type: none"> <li>• Build a sturdy robotic servo arm that has at least one degree of freedom.</li> <li>• Build a sturdy container to hold supplies.</li> <li>• Use LabVIEW to program their robotic arm to deliver supplies to the stranded mountaineers.</li> <li>• Use the engineering design process to solve the given challenge.</li> </ul>	<p>Evidence of learning found in:</p> <ul style="list-style-type: none"> <li>• A TETRIX arm that has at least 1 servo motor and a container to hold supplies.</li> <li>• Commented LabVIEW code.</li> <li>• A robotic arm that can deliver supplies.</li> <li>• Engineer's journal.</li> </ul>

# Lesson 4

# Reliable Reach

## Suggested Time

180 minutes

## Vocabulary

(See Appendix L)

Degrees of freedom  
Servo motor  
Potentiometer  
Pitch  
Yaw  
Roll

## Materials

### Each student:

- Engineer's Journal

### Each student group (4):

- LEGO MINDSTORMS kit
- TETRIX kit
- Computer with LabVIEW Education Edition

## Teacher Preparation

- Make copies of the Engineer's Journals.
- Select a weighted object for the robotic arm to lift. An ideal object would be something small and easy to incrementally increase the weight of (e.g. AA batteries, quarters, bean bags, etc.).
- Provide small flat-head and Phillips-head screwdrivers for motor assembly.

## Challenge

The mountaineers are stuck on a ledge that is not accessible to the rescue vehicle from the path below. As a result, the rescue vehicle will need to be equipped with a robotic arm that can assist with the delivery of supplies to the mountaineers. The students will need to design and build a robotic arm that has at least one **degree of freedom**, contain at least 1 **servo motor**, and can carry a specified minimum load.\* The robotic arm should be able to carry as much weight as possible and be able to hold the arm at a 45 degree angle. The students may want to experiment with the following arm parameters to maximize the amount of weight the arm can hold:

- Length of arm
- Gears
- Double servo motors

The students will also need to design a container to hold the supplies. They will construct the arm using the parts remaining in the TETRIX and NXT kits.

In this lesson the students will be focusing on the arm design, so they will prototype their robotic arm independent of the rescue vehicle chassis. They will need to build a base to attach the arm to. The final arm design will be attached to the rescue vehicle in Lesson 5.

*\* Set the minimum weight requirement based on the supplies available to you. For example, if you use AA batteries as weights you can specify that the arm needs to be able to hold a minimum of 4 batteries. Note that the students' container design is dependent on the type of weight you select.*

## Background

In a mechanical system, degrees of freedom (DOF) refer to the number of independent displacements or rotations in which motion is possible. For example, the human arm has seven degrees of freedom, a shoulder that allows for **pitch**, **yaw**, and **roll**, an elbow that allows for pitch, and a wrist that allows for pitch, yaw, and roll. The students will not be building a robotic arm as complicated as the human arm. Instead they will start out

## Lesson 4

## Reliable Reach

### Helpful Hint

Students can find examples of TETRIS robotic arms at <http://www.tetrisrobotics.com/inspiration/default.asp>

### Hardware Tip

DC motors have continuous rotation and can only do position control with an encoder and a control loop.

Servo motors usually have 180 degrees of motion and do position control without an encoder and external control loop.

### Classroom Management

The students can research robotic arms as a homework assignment in preparation for this activity.

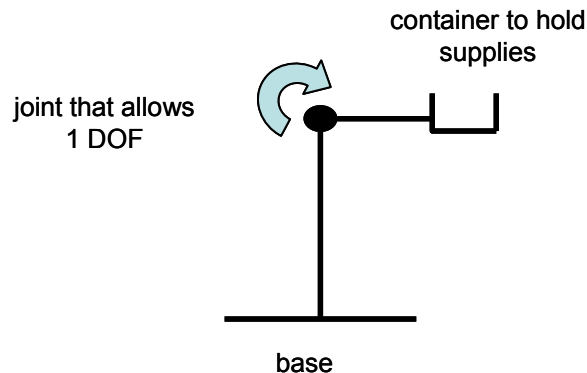
This activity works best if the students work in teams of 4 per TETRIS kit.

### Hardware Tip

Needle nose pliers are useful for tightening screws in hard to reach places.

Make sure the power switch is in the off position before connecting the servo motor set up to the DC motor controller.

with a simple one single degree of freedom robotic arm that is capable of moving in one direction (see below). Once the students master the single degree of freedom arm, they can increase complexity of the arm by adding additional degrees of freedom.



The students will use a servo motor to control the position of the robotic arm. Unlike a DC motor that requires an encoder and control loop to maintain position, a servo motor uses internal control (**potentiometer** and **pulse width modulation**) to maintain position. It is also important to note that the servo motor has a mechanical stop inside that prevents the motor from rotating more than 180 degrees.

## Instructions

### Part I: Introduction

**20 minutes**

1. Introduce the challenge to the students. Make sure they understand that they are prototyping a robotic arm and the final design will be attached to the chassis in Lesson 5.
2. Explain degrees of freedom and have the students analyze degrees of freedom in the human arm.
3. Ask the students to research robotic arms, especially focusing at degrees of freedom, design, and functionality. The students should note that most robotic arms have fewer degrees of freedom than the human arm.
4. As a class brainstorm some of the functionality that a robotic arm will need in order to deliver supplies to stranded mountaineers.
5. Introduce the servo motor and explain the difference between a DC motor and a servo motor.

## Lesson 4

## Reliable Reach

### Hardware Tips

The students will need a small flat-head screwdriver to connect wires from the DC motor controller terminals to the servo motor controller.

The students will also need a small Philips-head screwdriver to replace the plastic horn on the servo motor with a metal one.

Pay close attention to the color of the wires when connecting the servo motor to the servo motor controller. The servo motor has 3 wires; the yellow wire needs to be connected to the slot directly above the letter Y, the red wire above the letter R, and the black wire above the letter B. The red wire is power, black wire is ground, and the yellow wire is the signal.

### Programming Tip

The servo motor is located on the TETRIX palette.



### Part II: Design and Building

70 minutes

1. Depending on how comfortable you and/or your students are with the TETRIX construction set, you can either give your students the building instructions for the sample robotic arm (see Appendix I) or let your students design their own robotic arm. If the students are designing their own arm, have the students make a detailed drawing of their robotic arm design in their Engineer's Journal. The robotic arm must have at least one degree of freedom and have a container to hold the supplies. They should label their drawing. See Appendix F for the list of TETRIX component names.
2. Share the following TETRIX servo motor resources with your students:
  - a. Different ways of attaching the servo motor:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/servos\\_pivots.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/servos_pivots.pdf)
  - b. Daisy-chaining motor controllers:  
[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/labview/testbed/tetrix\\_testbed.html](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/labview/testbed/tetrix_testbed.html) (see section 6: connecting the servo controller)
3. Check the students' rescue vehicles and make sure that the arm and container are securely attached.
4. As the students finish building, tell them to draw their final design in their Engineer's Journal. They should label their drawing.
5. After the students are done with the building portion of the activity, have them share:
  - a. Their design with other teams.
  - b. Any building challenges they encountered and how they addressed them.

## Lesson 4

## Reliable Reach

### Hardware Tip

When the students are not testing the arm, turn off both the NXT brick and battery pack to save battery life.

### Tradeoffs

The longer the arm, the more torque is needed to hold up the same weight.

Students can increase the amount of torque supplied by the servo motor by either using 2 servo motors or using gears and attaching a small gear to the servo motor to drive a large gear. This will increase the amount of torque by decrease the precision. See [http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/basics/tetrix\\_primer/servos\\_pivots.pdf](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/basics/tetrix_primer/servos_pivots.pdf) for how to do this.

### Part III: Programming, Testing, Redesigning 70 minutes

1. Show the students how to program the servo motor. They will need to use the motor configurator to set up the servo motor port. For directions on how to program the servo motor see Appendix J.
2. Once the students figure out how to program their robotic arm to move to a position of 45 degrees, have them test out how much weight their robotic arm can hold.
3. Have them redesign their robotic arms and experiment with the following robotic arm parameters:
  - a. Length of arm
  - b. Gears
  - c. Double servo motors

How do these parameters affect the amount of torque and weight that the arm can support?

4. See which group's robotic arm can support the most weight at 45 degrees.

### Part IV: Class Discussion / Reflection 20 minutes

1. When all the students have completed the challenge, have the students present their design and demonstrate their final solution. Ask the students:
  - f. What programming challenges did they face?
  - g. What aspect of the challenge gave them the most trouble?
  - h. Did they redesign their robotic arm? If so, how did they redesign it?
2. Discuss the tradeoffs between:
  - i. Arm length vs. weight the arm can carry.
  - j. Gears: range of motion vs. amount of torque and precision?
3. Review the engineering design process and ask the students to identify which steps they completed and which ones they skipped. How might the skipped steps have been helpful?

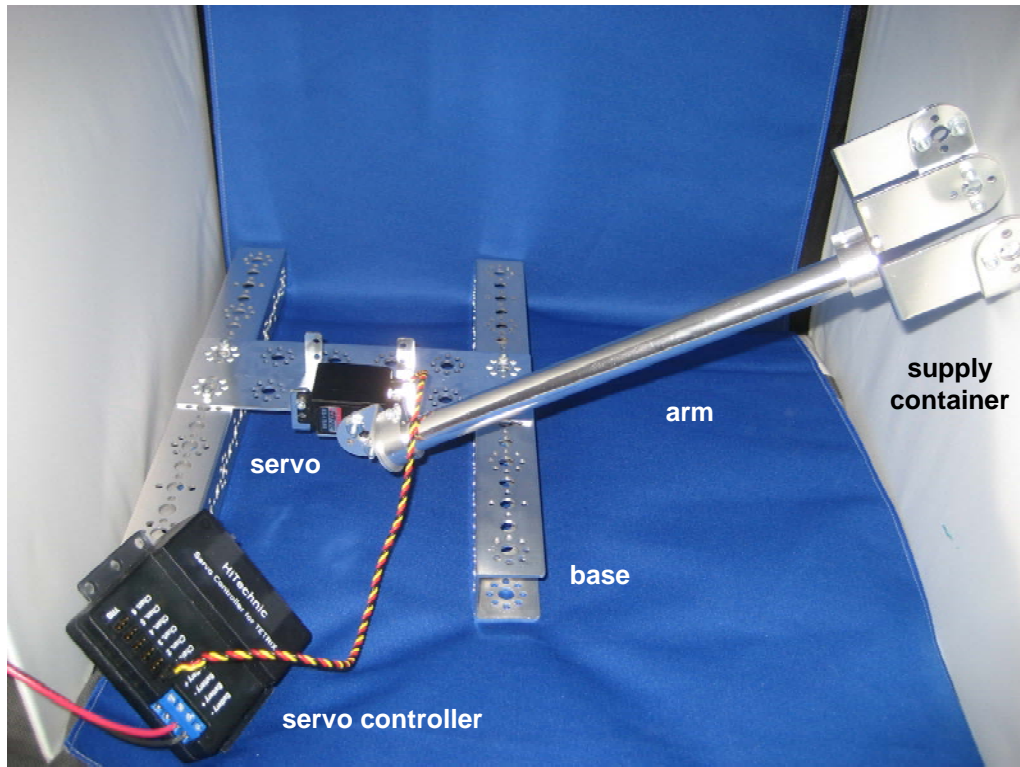
### Extensions

1. Programming: Write a program that will allow you to control the servo motor position using an NXT motor joystick.
2. Building: Build a robotic arm that has two or more degrees of freedom (can use NXT motors).
3. Teamwork: Pair up with another team to create a joystick controlled robotic arm using Bluetooth (need two NXTs).

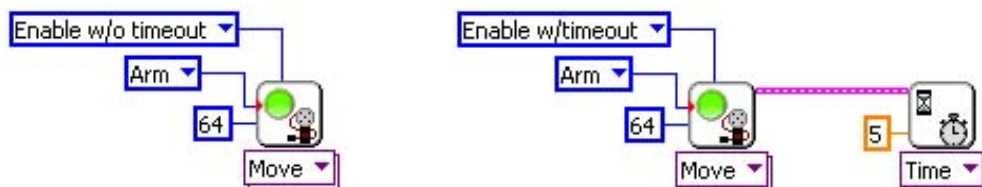
## Sample Project and Photos

---

Photo of sample TETRIX robotic arm:



Sample code:



Servo motor without timeout

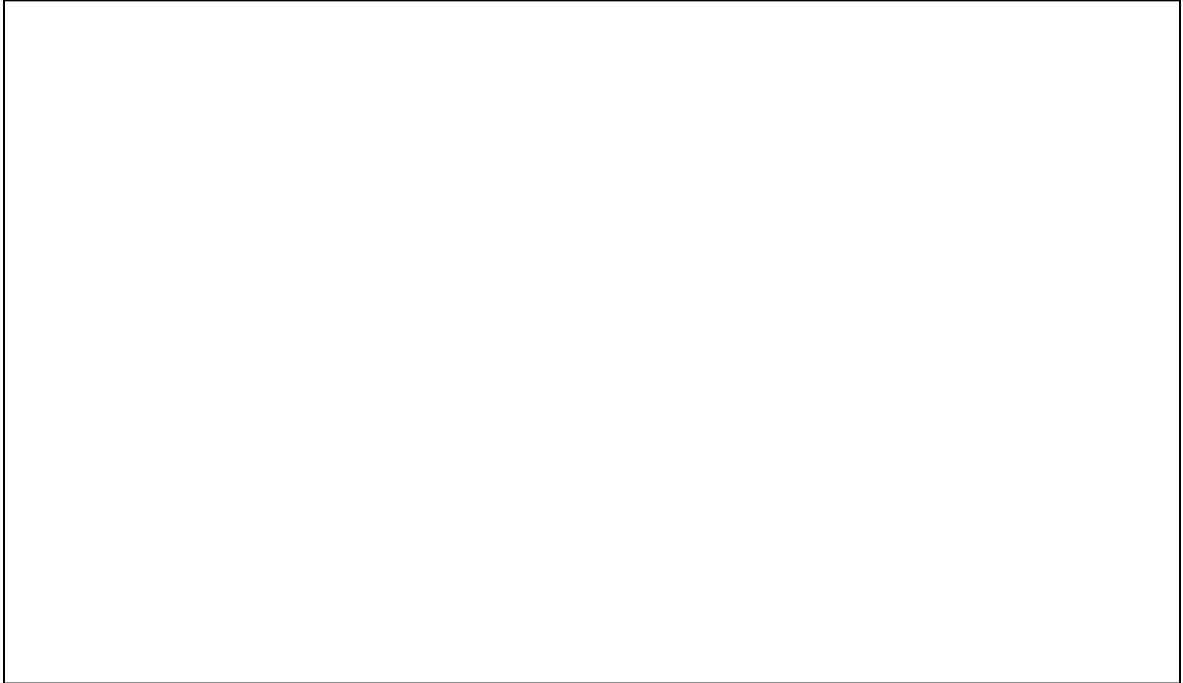
Servo motor with timeout

\* See Appendix J for more information on programming the servo motor.

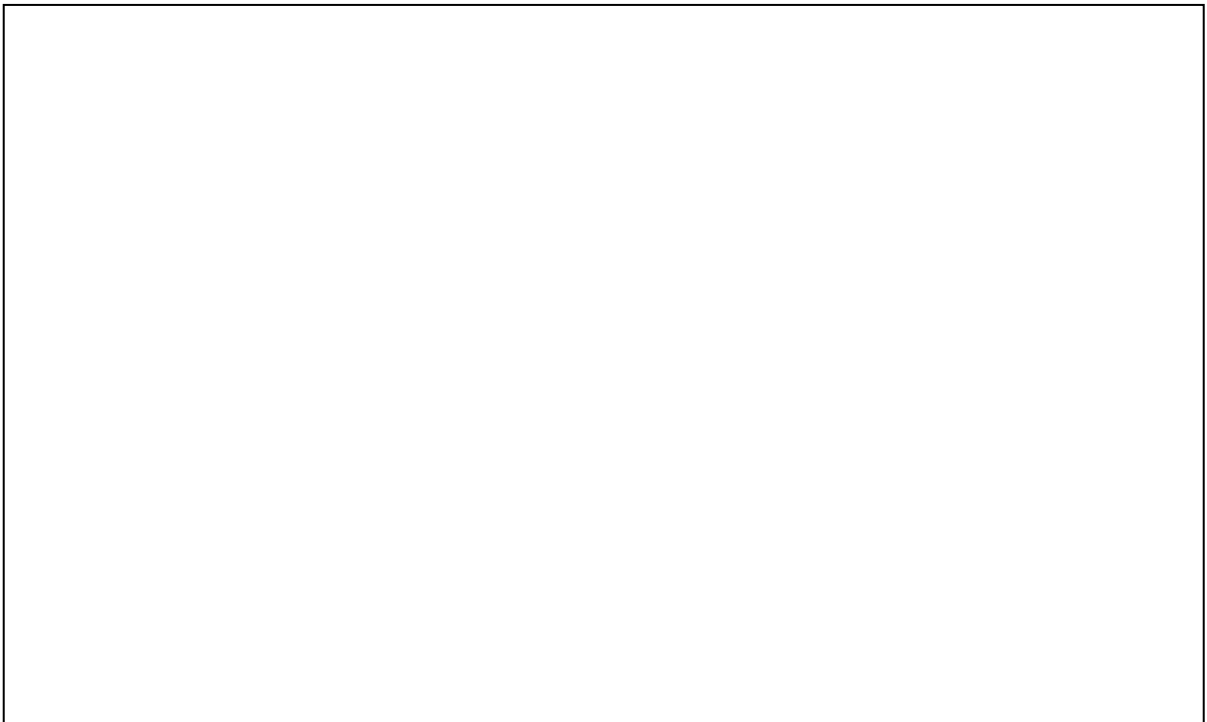
## Lesson 4 Engineer's Journal

### Reliable Reach

1. Make a detailed drawing of the TETRIX arm that you plan to build. Label your drawing (servo motor, gears, container, etc.)



2. Sketch your final robotic arm and label the components. Label your drawing and include dimensions. Also include multiple views if possible.





3. How much weight was your robotic arm able to hold?

---

4. What parameters did you modify on your robotic arm to increase the amount of weight that it can carry?

---

---

---

---

5. What are some of the challenges that you encountered in this activity?

---

---

---

---

6. What did you learn from this activity?

---

---

---

---

7. What steps of the engineering design process did you use?

---

---

---

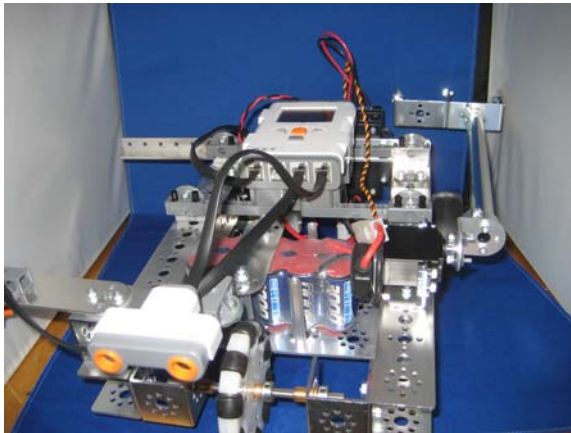
---

# Ultimatum

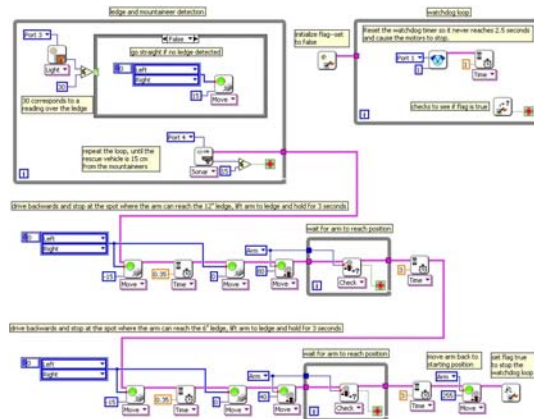
## Overview

In this final challenge the students will deploy their TETRIX rescue robot up the mountain to rescue the stranded mountain climbers. First the rescue robot has to navigate up the mountain, and then deliver supplies to the stranded mountaineers located on ledges of two different heights. The students will review the Pythagorean Theorem and trigonometry to determine the position of the rescue robot and angle of the supply delivery arm to successfully reach the specified target heights.

Sample TETRIX Rescue robot



Sample Code



Expectations	Evidence
<p>Students should be able to:</p> <ul style="list-style-type: none"> <li>• Attach the TETRIX robotic arm (from Lesson 4) to the chassis (from Lesson 3).</li> <li>• Use LabVIEW to program the rescue robot to navigate to the stranded mountaineers and reach two target heights.</li> <li>• Use the engineering design process to solve the given challenge.</li> </ul>	<p>Evidence of learning found in:</p> <ul style="list-style-type: none"> <li>• A TETRIX rescue robot with an arm securely attached to the chassis.</li> <li>• Commented LabVIEW code.</li> <li>• A supply delivery arm that can reach two target heights.</li> <li>• Engineer's journal.</li> </ul>

# Lesson 5

# Ultimatum

## Suggested Time

180 minutes

## Vocabulary

(See Appendix L)

Pythagorean Theorem  
Hypotenuse

## Materials

### Each student:

- Engineer's Journal

### Each student group (4):

- LEGO MINDSTORMS kit
- TETRIX kit
- Computer with LabVIEW Education Edition
- Ruler and protractor

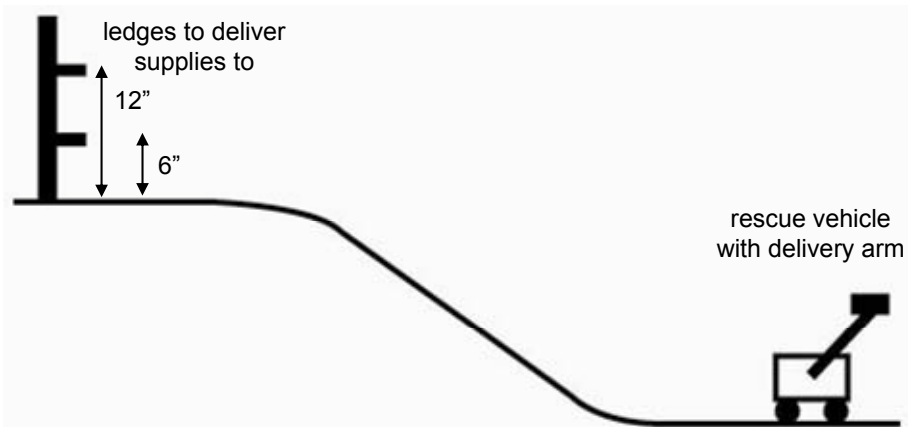
## Teacher Preparation

- Make copies of the Engineer's Journals.
- Provide small flat-head and Phillips-head screwdrivers for DC and servo motor assemblies.
- Build a wall with ledges at 6" and 12" for the rescue robots to reach. The wall can be constructed from boxes, LEGOs, books, etc. If you want the supply delivery arm to unload the supplies, make sure that the ledge is capable of holding some weight. Also make sure that the wall can fall over if the rescue robot crashes into it. This will minimize the damage to the rescue robot.
- Set-up course where the challenge will take place.

## Challenge

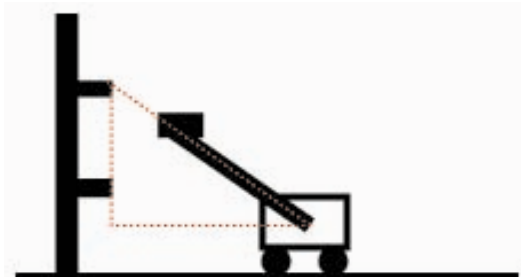
It's time to assemble your entire rescue robot and drive up the mountain to deliver supplies to the stranded mountaineers! In addition to navigating up the mountain and stopping upon arrival at the destination, your rescue robot also has to be able to deliver supplies to mountaineers stuck on two different ledges of different heights: 6" and 12" off the ground (see below). Program the rescue robot to hold the supply delivery arm up to the stranded mountaineers so that they can unload the supplies. The rescue robot should avoid collision with the wall.

For a more difficult challenge, build a two degree of freedom robotic arm and program your rescue robot to deliver *and* unload supplies onto the ledge for the stranded mountaineers.



## Background

The students will need to use the **Pythagorean Theorem** and trigonometry to determine where the rescue robot needs to be and what angle to set the supply delivery arm so that it can reach the target.



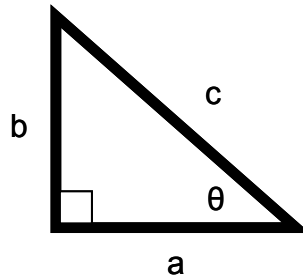
## Lesson 5

## Ultimatum

### Real World Connection

Sometimes it is not possible for engineers to measure every single parameter they are interested in without affecting the experiment. Just like you infer how far the rescue robot travels based on the motors' power setting and time, engineers often infer the information they want based on what is known.

As you can see on the previous page, the robot's arm is the hypotenuse of the right triangle that is formed between the robot arm, the wall, and the space between the wall and the rescue robot. The students can measure the length of the supply delivery arm and the height of the wall. Using this information they can calculate the unknown parameters:  $a$  and  $\theta$ .



Pythagorean Theorem:

$$a^2 + b^2 = c^2$$

Trigonometry:

$$\sin(\theta) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{b}{c}$$

#### Determining servo position:

1. Measure the length of the arm ( $c$ ).
2. Calculate the vertical distance from the height of the supply delivery arm pivot to the ledge ( $b$ ).
3. Use trigonometry to determine  $\theta$ .
4. Convert  $\theta$  to position. (Note: 0-180 degrees corresponds to a position of 0-255.)

#### Determining position of rescue robot:

1. Use Pythagorean Theorem to determine how far the supply delivery arm pivot needs to be from the wall ( $a$ ).
2. Since the TETRIX kit does not come with encoders, the students will need to infer the distance the rescue robot travels based on time (for a given DC motor power setting). As in Lesson 1, the students can plot the distance travelled versus time to determine the relationship between the two.

*Note: Do not use the "Wait for Away" function because there is a bug in function and does not work correctly.*

## Lesson 5

## Ultimatum

### Building Tip

The students may need to modify their robotic arm design when they interface the arm with the chassis.

### Math Tip

If the students built their rescue robots based on the sample build instructions provided in Lessons 3-5, then the rescue robot (and the arm) is not perfectly level. The rescue robot will have a 10 degree incline. The students will need to add 10 degrees to their calculated angular position when they write their program.

All the calculations should be made from the supply delivery arm pivot point. The students should subtract the height of the pivot point from the height of the ledge.

## Instructions

---

### Part I: Introduction

10 minutes

1. Introduce the challenge to the students.
2. Show them the course that the rescue robot has to navigate to complete the challenge.

### Part II: Building

30 minutes

1. Depending on how comfortable you and/or your students are with the TETRIX construction set, you can either give your students the building instructions for how to attach the sample robotic arm to the sample rescue robot chassis (see Appendix K) or let your students figure out how to attach the robotic arm they designed and built in Lesson 4 to the chassis they built in Lesson 3.
2. After the students are done attaching the arm, ask them to draw their final design in their Engineer's Journal. The students should label their drawing and provide short descriptions of the functionality of the major parts of their rescue robot. See Appendix A for the list of TETRIX component names.

### Part III: Calculations

20 minutes

1. If your students have not yet learned the Pythagorean Theorem or trigonometry (or if they need a math review) show them how they can use these mathematic tools to determine the unknown parameters: distance the arm pivot needs to be from the wall and the angle the arm needs to be held at.
2. Have the students calculate the distance the rescue robot has to travel and the angle the arm needs to be held at to reach the following target heights: 6 inches, 12 inches.
3. After the students calculate the angular position of the arm, they should convert the angles to position (note: 0–180 degrees correspond to a position of 0-255 on the servo motor). For more information on the HiTechnic servo motors, read the servo motor context and detailed help.

## Lesson 5

## Ultimatum

### Programming Tip

Remember to include the watchdog loop if you want to run your DC motors for more than 2.5 seconds.

### Classroom Tip

The performance of the robot may vary depending on battery level. Make sure that both the TETRIX and NXT battery packs are fully charged.

### Activity Tip

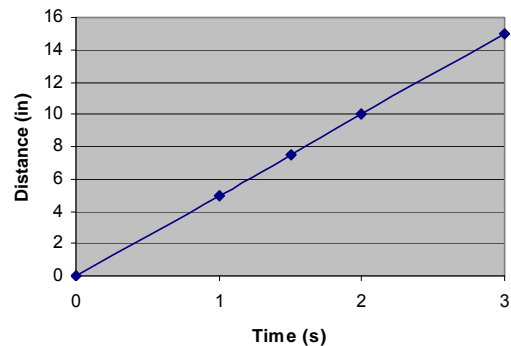
The calculated wait for time and servo arm position will serve as a good first estimate of what values the students should input into their rescue robot program. The students will most likely have to adjust their values to get the supply delivery arm to the actual ledge. Some of the factors that can result in a difference between the calculated values and the actual values are:

- Inertia
- Environmental factors (e.g. surface the car is on)
- Battery power

### Part IV: Programming, Testing, Redesigning 100 minutes

1. First the students need to write a program that will allow them to determine how far the robot drives per unit of time. The students should write a short program that will allow them to turn on the HiTechnic DC motors for a set amount of time. Encourage the students to keep track of time versus distance in a table as shown below. Then, students can create a graph with this information to find the exact time needed to travel the distances they determined in Part III.

Time (s)	Distance (in)



2. As a class, make a flow diagram of the program for the rescue robot. For example, the rescue robot has to:
  - a. Navigate up the ledge without falling off the ledge.
  - b. Stop when it reaches the mountaineers.
  - c. Deliver supplies to mountaineers on one ledge.
  - d. Deliver supplies to the mountaineers on the other ledge.
3. Once the students have a flow diagram of their program, they can go ahead and program their robot to complete the rescue mission.
4. Have the students test their programs and debug the code until they have a robot that can repeatedly accomplish the mission.
5. The students should add comments to the code to document their work.

## Lesson 5

## Ultimatum

### Activity Tip

The calculated wait for time and servo arm position will serve as a good first estimate of what values the students should input into their rescue robot program. The students will most likely have to adjust their values to get the supply delivery arm to the actual ledge. Some of the factors that can result in a difference between the calculated values and the actual values are:

- Inertia
- Environmental factors (e.g. surface the car is on)
- Battery power

### Part V: Class Discussion / Reflection

**20 minutes**

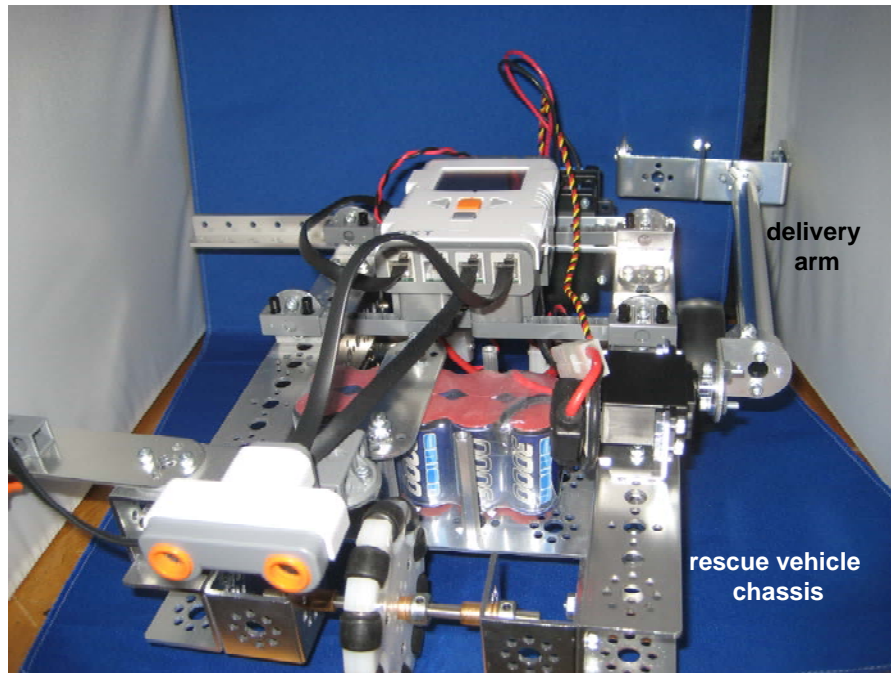
1. When all the students have completed the challenge, have the students present their design and demonstrate their final solution. Ask the students:
  - k. What programming challenges did they face?
  - l. What aspect of the challenge gave them the most trouble?
  - m. Did they redesign their robotic arm? If so, how did they redesign it?
2. Ask the students to compare their calculated values to the actual values they used. What could have caused the difference? Discuss.
3. Review the engineering design process and ask the students to identify which steps they completed and which ones they skipped. How might the skipped steps have been helpful?

### Extensions

1. Programming: Program the robot to maneuver around unpredictable objects it may encounter on the path.
2. Building: Build a rescue robot that can climb up stairs (instead of a flat or inclined surface).
3. Teamwork: Work with another group on transferring supplies between robotic arms.

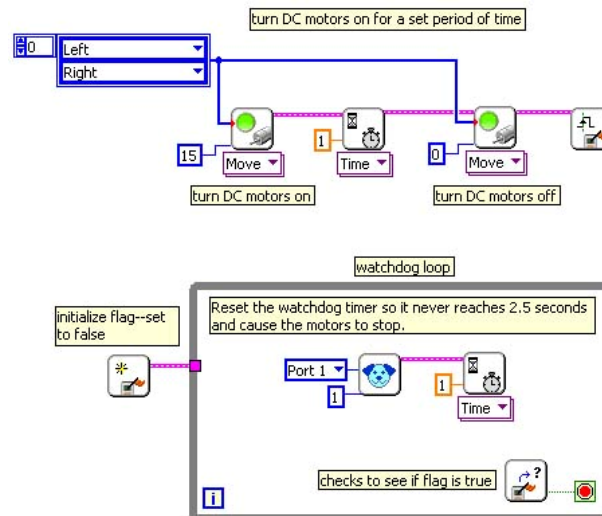
## Sample Project and Photos

Photo of sample TETRIX rescue robot:



Sample code:

This program turns the DC motors on for a set period of time. It can be used to determine how far the robot travels per unit of time.

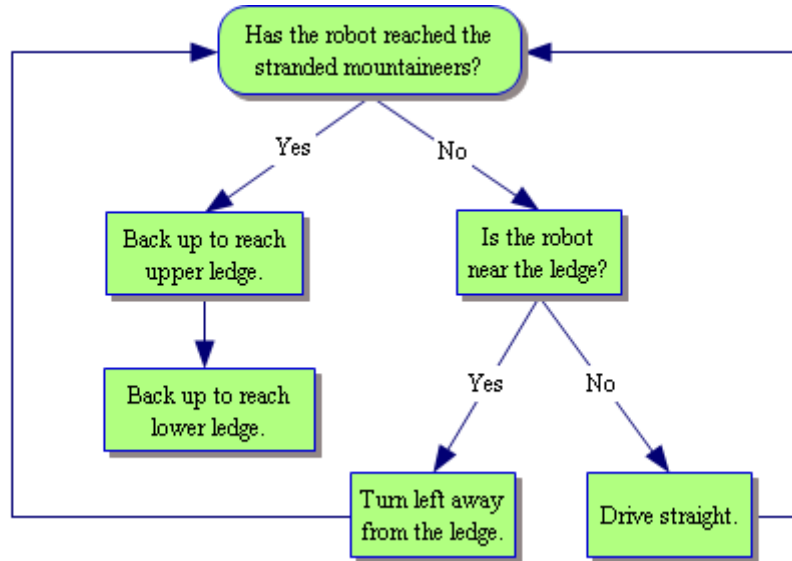




## Sample Project and Photos

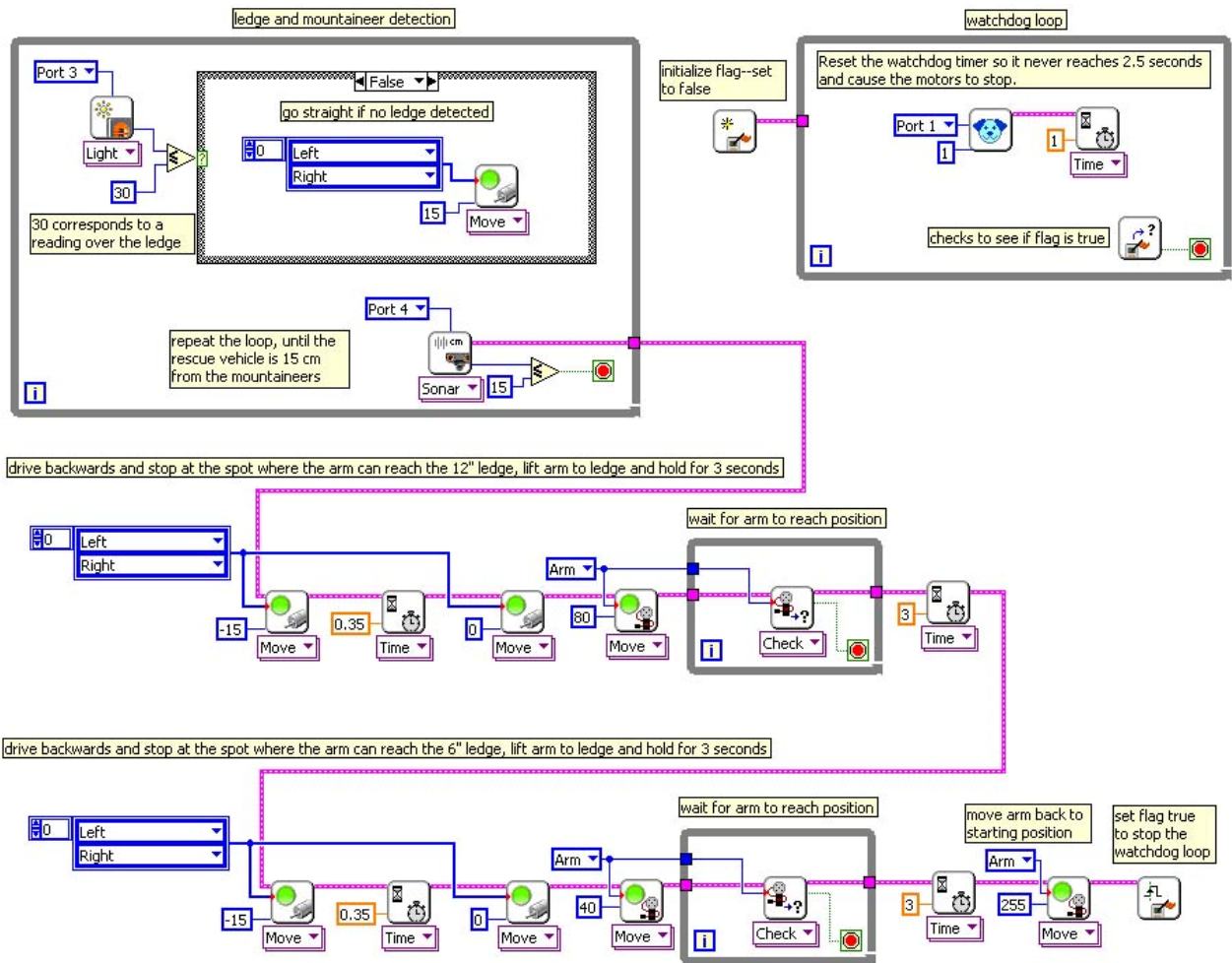
---

This is a sample flow diagram of the code:



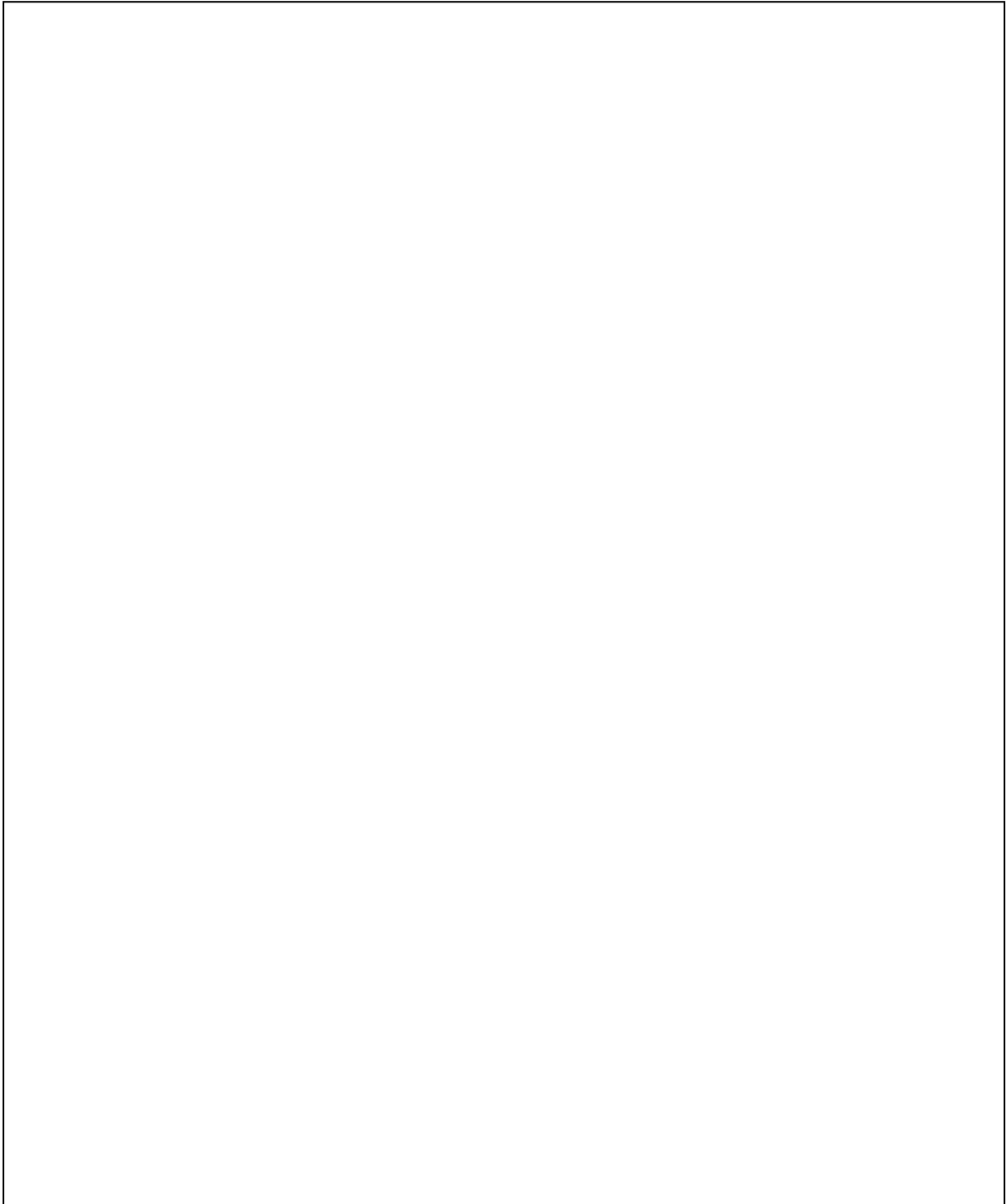
### Sample Project and Photos

In the following sample code, the rescue robot first executes ledge and mountaineer detection. Once the rescue robot knows it has arrived at the designated location, it backs up to the 12" ledge mark, lifts the arm up, waits 3 seconds, backs up to the 6" ledge, lowers the arm, waits 3 seconds, and then returns the arm back to the resting position.



## Lesson 5 Engineer's Journal Ultimatum

1. Sketch your final rescue robot and label your drawing (chassis, DC motor, servo motor, gears, arm, container, etc.). Also include a short description of the functionality of each part.



2. Calculate the distance your rescue robot has to travel and the degrees the arm needs to be held at to reach the following target heights: 6 inches, 12 inches. Show your math and include units.

Target height = 6 inches

Horizontal distance the arm pivot needs to be from the wall.	Angle the arm needs to be at to reach the target.  Convert the angle to a servo position (0-255).
--	---

Target height = 12 inches

Horizontal distance the arm pivot needs to be from the wall.	Angle the arm needs to be at to reach the target.  Convert the angle to a servo position (0-255).
--	---

3. For a DC motor power setting of 15, calculate the exact time it takes for the rescue robot to travel the distances that you calculated in the previous step.

Target height = 6"	Target height = 12"
Horizontal distance =	Horizontal distance =
Time =	Time =

4. Draw a flow diagram of the program you will write to complete the challenge in this lesson.



5. What are the actual distances your rescue robot travelled and the degrees the arm needs to be held at to reach the target heights 6 inches and 12 inches? Include units.

Target height = 6"	Target height = 12"
Actual horizontal distance the arm pivot needed to be from the wall =	Actual horizontal distance the arm pivot needed to be from the wall =
Actual angle the arm needed to be at to reach the target =	Actual angle the arm needed to be at to reach the target =

6. What was the difference between the calculated values and the actual values used? Include units.

Target height = 6"	Target height = 12"
Distance difference =	Distance difference =
Angle difference =	Angle difference =

7. What factors might have caused the difference between the calculated values and the actual values?

---



---



---



---

8. What are some of the challenges that you encountered in this activity?

---

---

---

---

9. What did you learn from this activity?

---

---

---

---

10. What steps of the engineering design process did you use?

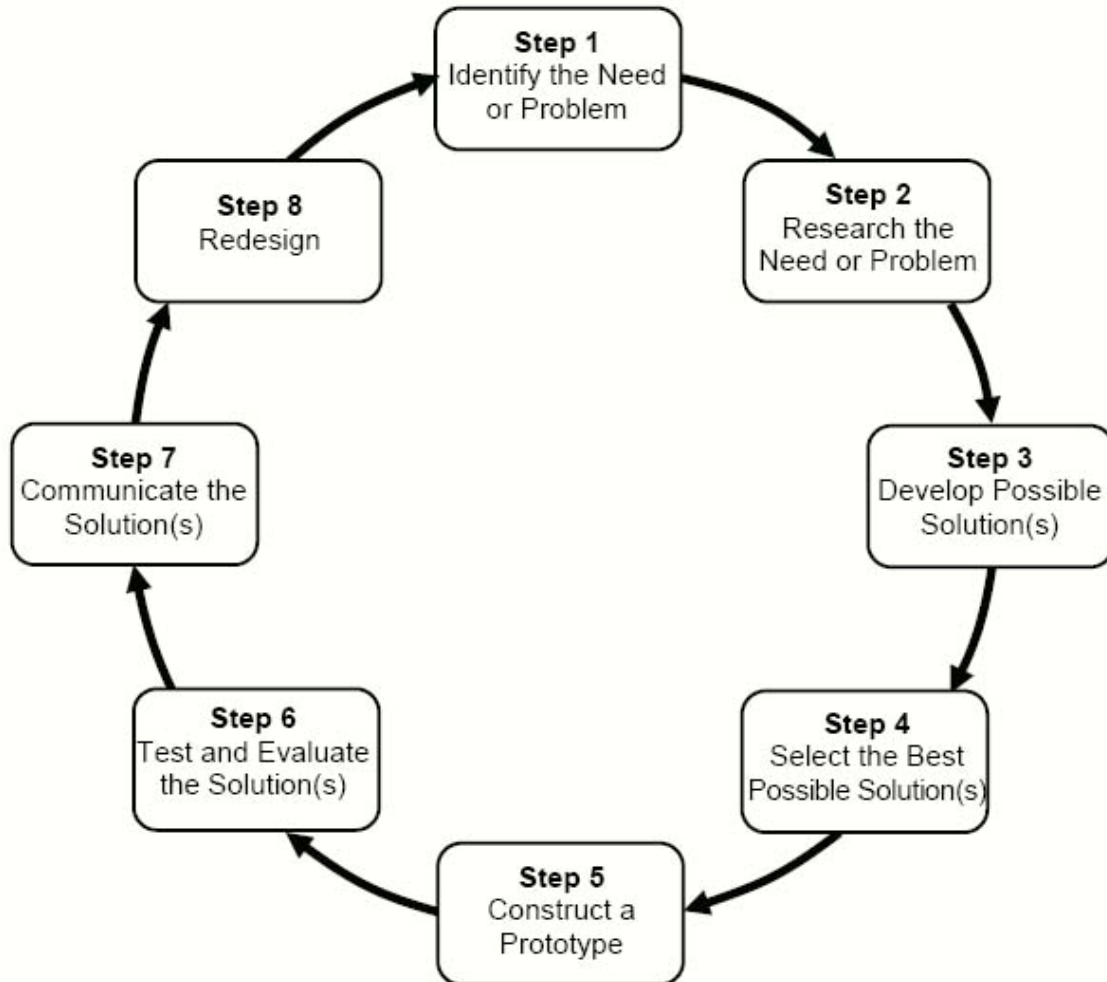
---

---

---

---

Massachusetts Science and Technology/Engineering Curriculum Framework,  
October 2006



1. Identify the need or problem.
2. Research the need or problem
  - a. Examine the current state of the issue and current solutions
  - b. Explore other options via the internet, library, interviews, etc.
3. Develop possible solution(s)
  - a. Brainstorm possible solution(s)
  - b. Draw on mathematics and science
  - c. Articulate the possible solution(s) in two or three dimensions
  - d. Refine the possible solution(s)
4. Select the best possible solution(s)
  - a. Determine which solution(s) best meet(s) the original need or solve(s) the original problem



5. Construct a prototype
  - a. Model the selected solution(s) in two and three dimensions
6. Test and evaluate the solution(s)
  - a. Does it work?
  - b. Does it meet the original design constraints?
7. Communicate the solution(s)
  - a. Make an engineering presentation that includes a discussion of how the solution(s) best meet(s) the initial need or the problem
  - b. Discuss societal impact and tradeoffs of the solution(s)
8. Redesign
  - a. Overhaul the solution(s) based on information gathered during the tests and presentation

### Axle

- A rod which is used to house wheels and join beams together.



### Beam

- A building unit of LEGOs with holes through the sides. The typical building unit of the NXT Mindstorms kits.



### Bushing

- A small piece which fits over the ends of axles to secure them into place.



### Input

- Take in information about the surrounding environment.  
e.g. - NXT sensors.  
e.g. - Eyes, nose, etc.

### Motor

- A geared engine that provides rotation when supplied with electricity. A motor converts electricity into mechanical movement.



### NXT

- A computer controlled brick and the “brain” of your robot that controls the motors and sensors.



### Output

- An action that reacts to the environment.  
e.g. - A motor which turns on and off according to the NXT “brain”.  
e.g. - Swinging a baseball bat when your brain tells you to.

### Peg

- A LEGO piece that goes through the holes in the beams.

#### Friction Peg:

- A peg that prevents the beams from moving when connected.



### Port

- The place on the NXT where the input and output wires are inserted into the NXT. They are labeled 1, 2, 3, and 4 for the input and A, B, and C for the output.

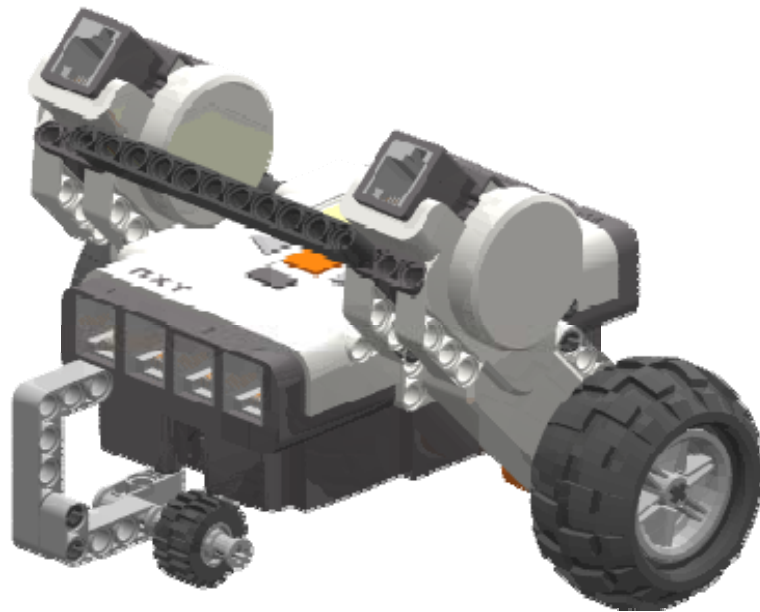
### Wire

- The method of transport of electricity for the motors and sensors.

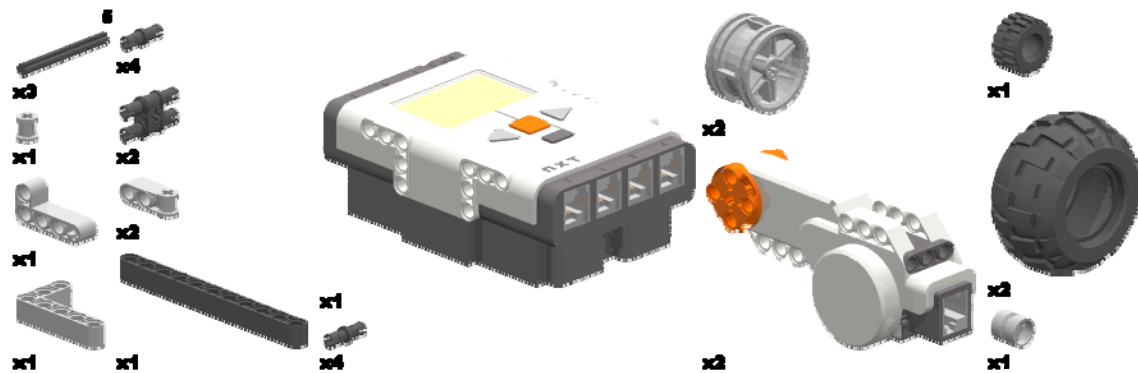


# Appendix C

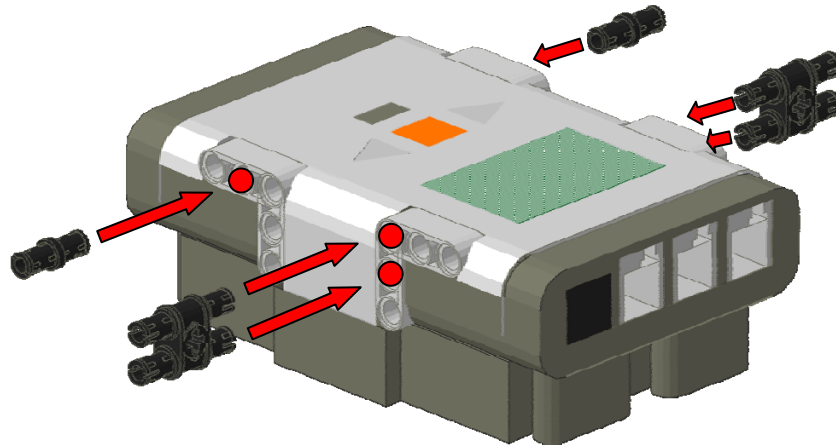
Two motor car:



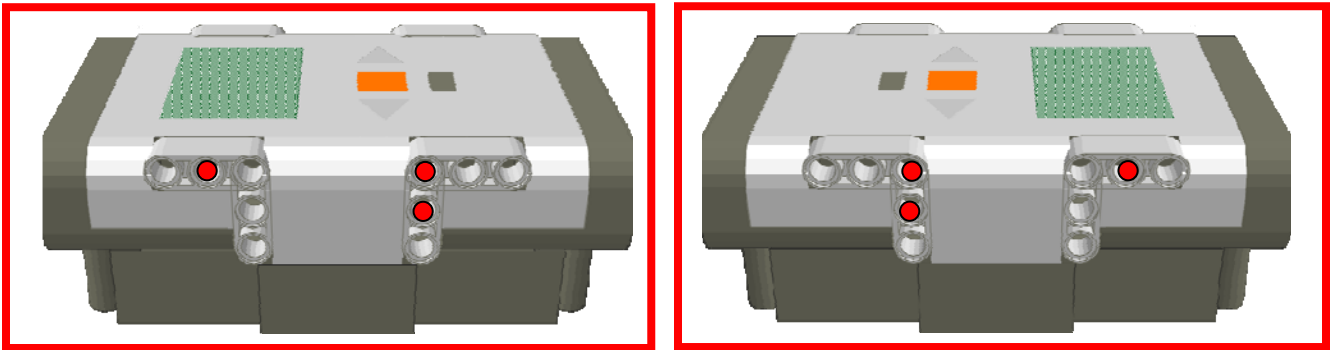
Required pieces:



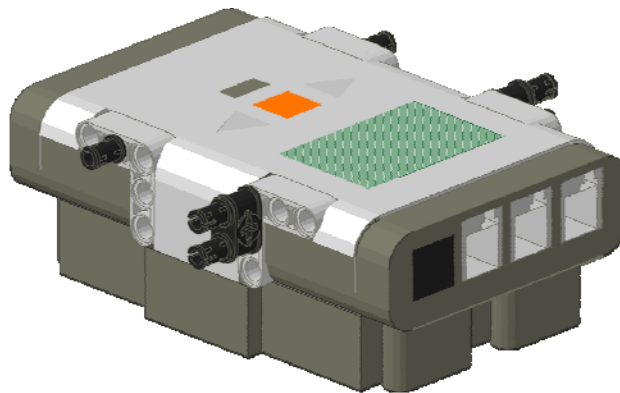
## Step 1: Prepare NXT



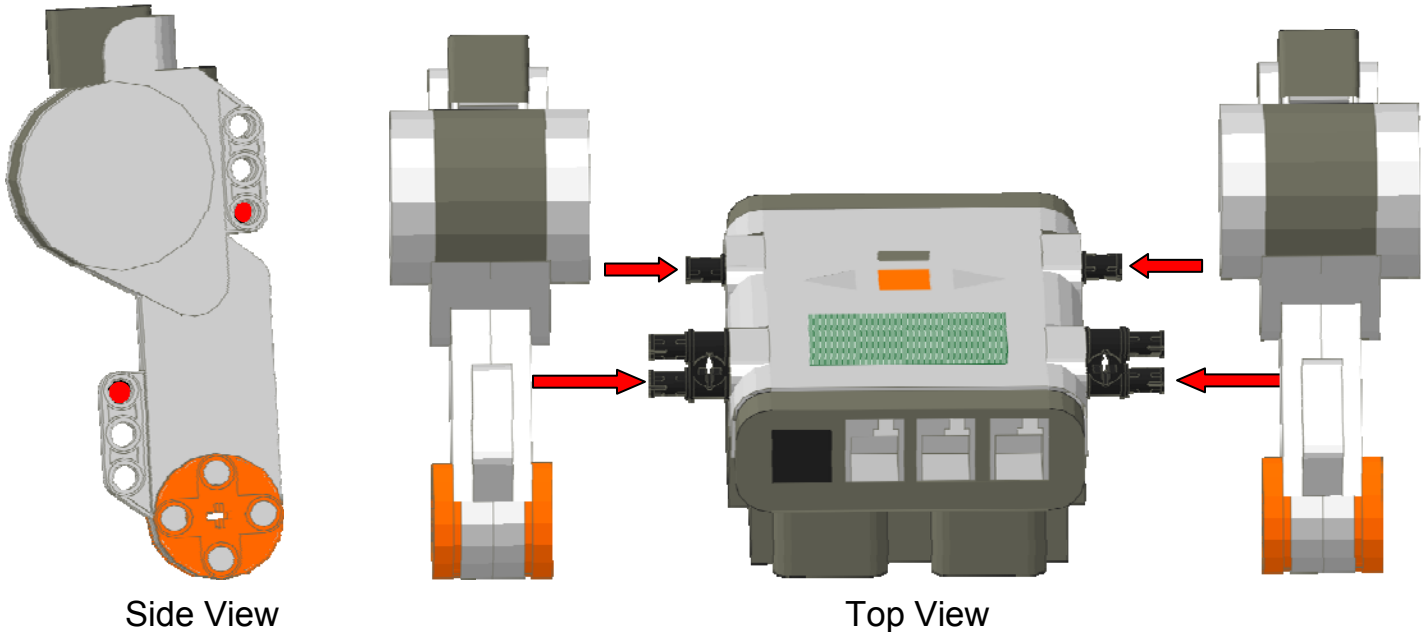
## Side Views



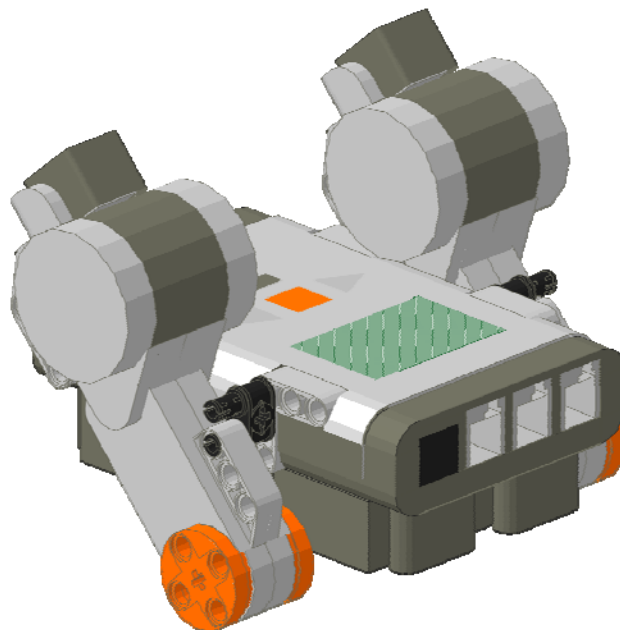
Attach a double black connector peg and a single black connector peg to both sides of the NXT. Connect the double peg vertically in the top most holes. The single peg should be connected in the middle horizontal hole. Your NXT should look like the below picture before moving on to the next step.



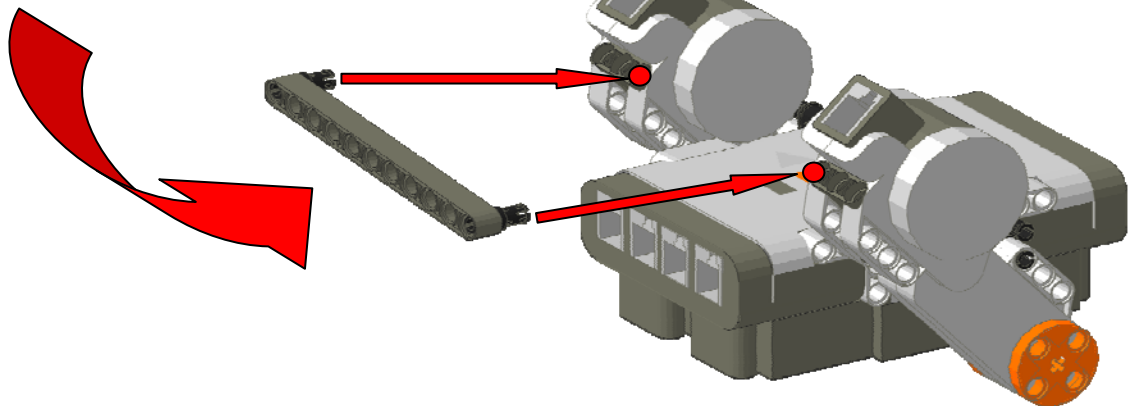
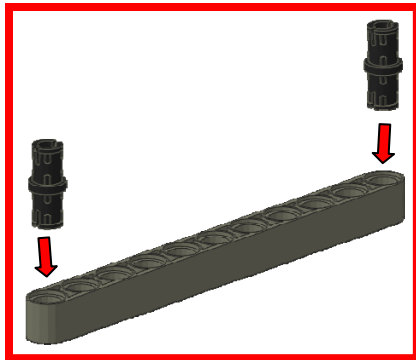
## Step 2: Attach Motors to the NXT



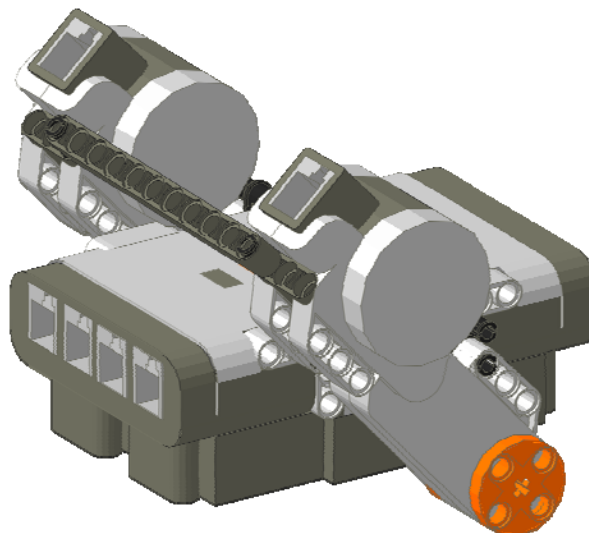
Attach the motors (one per side) to the NXT using the double black connector pegs and the short black connector pegs attached to the NXT. The **red** dots in the side view identify which holes on the motor attach to the pegs. Your NXT should look like the below picture before moving on to the next step.  
Note: The top connection of the double black connector peg is not connected to anything.



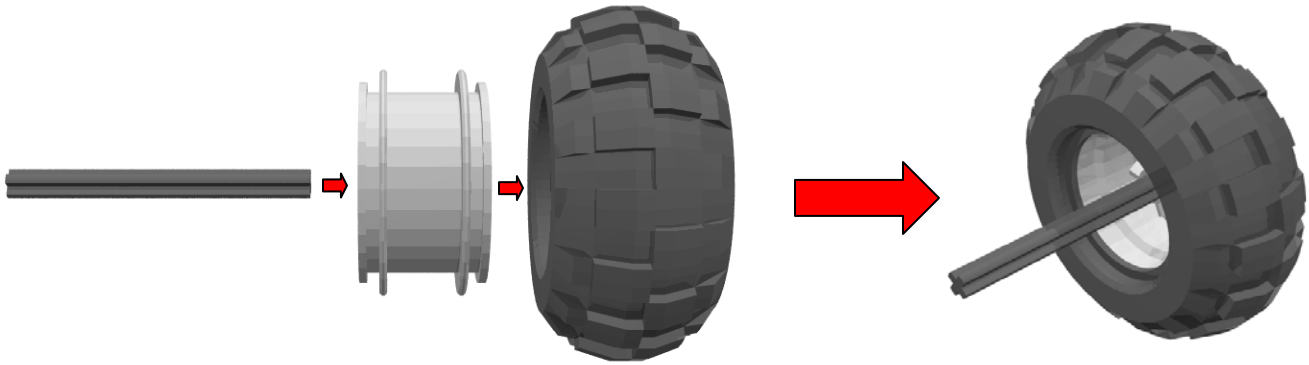
## Step 3: Support Motors



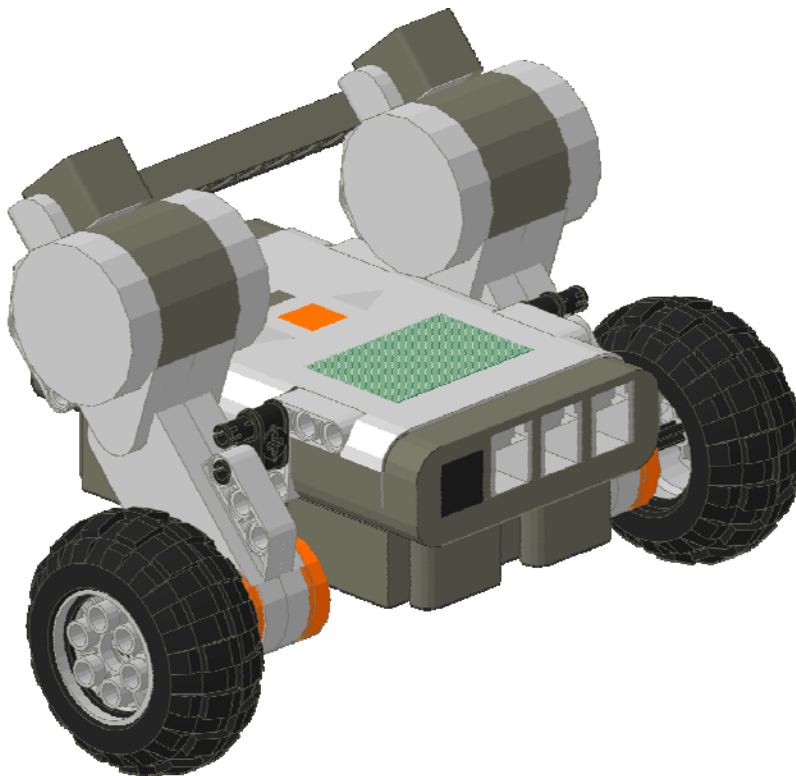
Take an 11-holed rounded beam and attach two short black connector pegs to the outside holes as seen in the **red** square. Connect the pegs to the back of each motor (as identified by the **red** dots) to further support the motors to the NXT. Your NXT should look like the below picture before moving on to the next step.



## Step 4: Attach Wheels

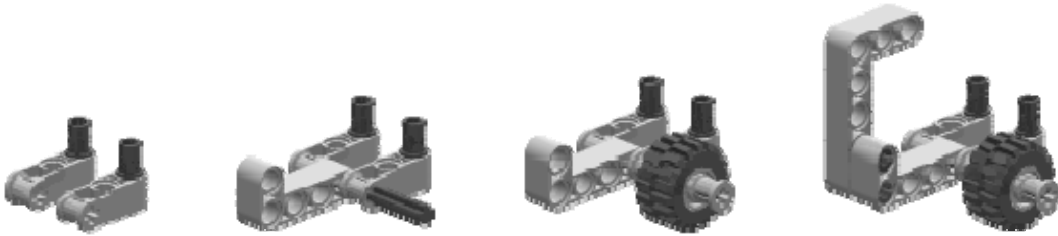


Assemble 2 rear wheel assemblies using a 6-length axle, a wheel, and a hub. Attach one to each motor as seen below.

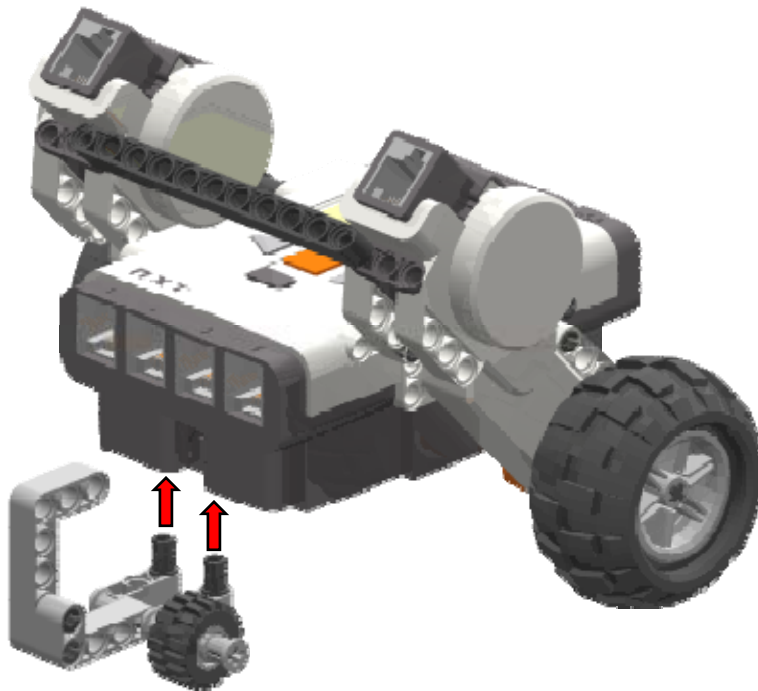




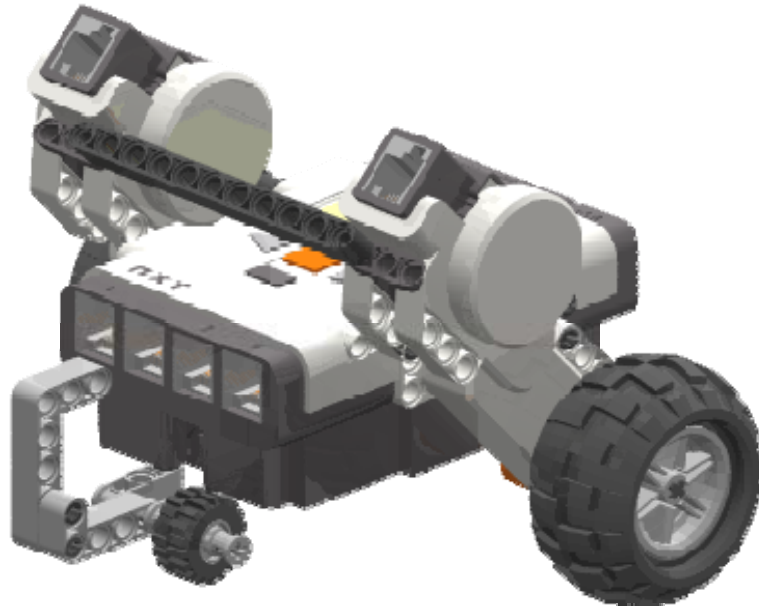
Step 5: Front Wheel Base Assembly



Step 6: Front Wheel Assembly Attachment



Finished NXT Vehicle:



**Step 1:** Turn on the NXT by pressing the orange select button on the NXT brick. The NXT will turn on and show the main menu.



**Step 2:** Using the gray arrow navigation buttons, navigate through the menus until you see the view icon.



**Step 3:** Select “View” by pressing the orange select button on the NXT brick. You will be directed to a list of sensors. Navigate to the “Reflected Light” icon using the gray arrow buttons. Select this menu option by pressing the orange select button.



**Note:** Reflected light measures the amount of light that is returned to the light sensor from light that the LED on the light sensor generates. Ambient light measures the light levels in the environment around the light sensor. The “Ambient Light” option is what you should use when you want to know about light levels surrounding the NXT. The “Reflective Light” option is what to use to measure light reflected off of materials (darker, less shiny materials will not reflect much light, whereas lighter, more reflective materials will reflect a lot of light).

**Step 4:** You will be directed to a menu of ports. Navigate to the correct port using the gray arrow buttons. Select the port that your sensor is connected to using the orange select button.

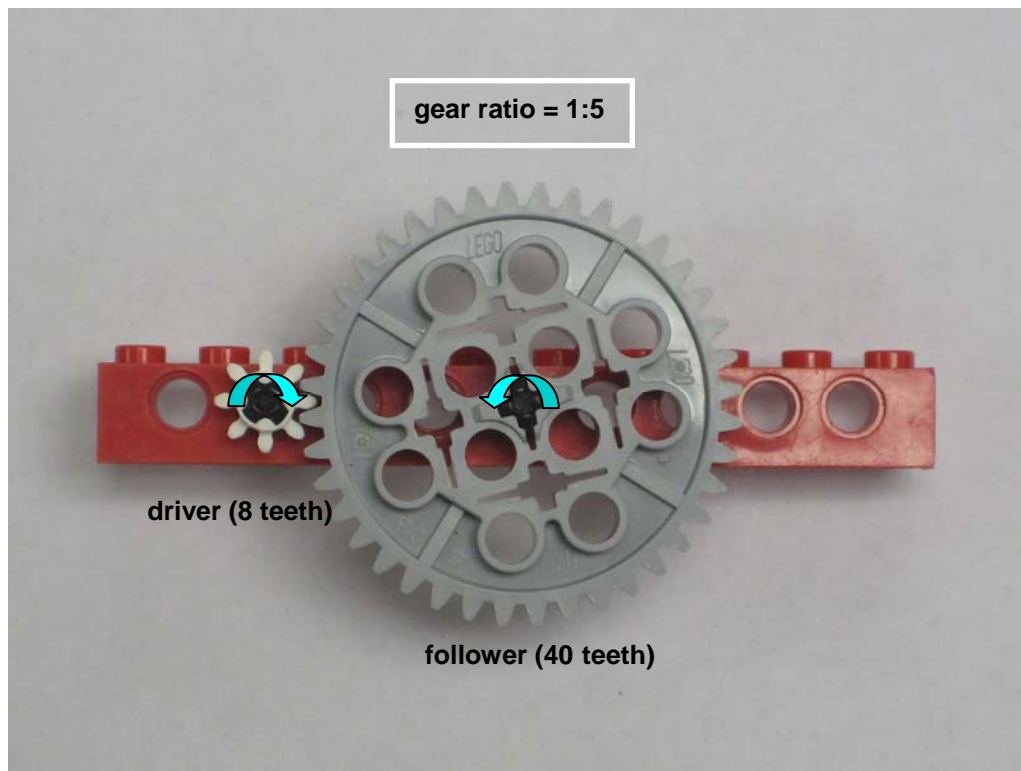


**Step 5:** At this point, you will be able to view the light levels that your light sensor is measuring. The NXT measures light levels on a scale of 0 – 100, with 0 being the darkest, 100 being the lightest.



Gears are used to transmit motion and force. When gears of different sizes are paired together, the larger gear will provide more force, but move slower than the smaller gear. Each gear rotates in the opposite rotational direction from the previous gear. The gear that is powered is called the driver. The last gear in a gear train is called the follower. The gears between the driver and follower are called idle gears.

A gear ratio is used to determine how many times one gear will turn in relation to another. The gear ratio is determined by the number of teeth, or cogs, on the drive gear compared to the number of teeth on the follower (e.g. driver with 8 teeth, follower with 40 teeth = gear ratio of 1:5).



# TETRIX™ by Pitsco

## 2009 Product Images

**SINGLE-SERVO BRACKET**

Bracket for the placement of a single servo

**32 MM CHANNEL**

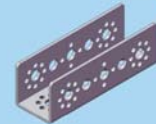
Structural element for building heavy-duty bases and building components

**FLAT BRACKET**

Perfect for connecting structural elements or for creating a servo leverage arm

**96 MM CHANNEL**

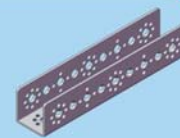
Structural element for building heavy-duty bases and building components

**L BRACKET**

90-degree angle bracket that enables bends and angled components

**160 MM CHANNEL**

Structural element for building heavy-duty bases and building components

**SERVO JOINT PIVOT BRACKET**

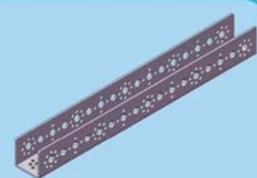
Enables pivoting arms, grippers, and appendages

**PIVOT BEARING**

Pivot bearing for the Servo Joint Pivot Bracket

**288 MM CHANNEL**

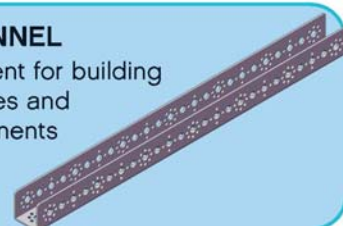
Structural element for building heavy-duty bases and building components

**DOUBLE-SERVO BRACKET**

Bracket for the placement of two servos

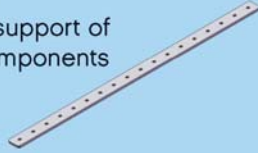
**416 MM CHANNEL**

Structural element for building heavy-duty bases and building components

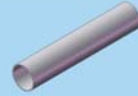


**288 MM FLAT BAR**

Enables strength and support of bases and building components

**80 MM TUBE**

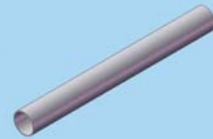
Structural element used when needing extensions such as arms and grippers

**288 MM ANGLE**

Large Angle, which enables bends and angled components and reinforcement of a structure

**145 MM TUBE**

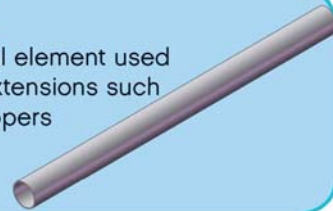
Larger structural element used when needing extensions such as arms and grippers

**144 MM ANGLE**

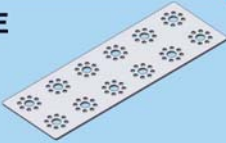
Small Angle, which enables bends and angled components and reinforcement of a structure

**220 MM TUBE**

Largest structural element used when needing extensions such as arms and grippers

**FLAT BUILDING PLATE**

A baseplate ideal for larger mounting areas for electronics and other TETRIX™ components

**TUBE PLUG**

Reinforces tubes when applying a tube clamp

**SET SCREW AXLE HUB**

Used to attach gears and wheels to axles and axles to structural elements

**TUBE CLAMP**

Used to attach tubes to structural elements


**MOTOR HUB**

Attaches wheels and gears to TETRIX™ DC Motor shafts

**SERVO WITH HORN**

Lightweight servo motor that enables 180-degree movement in a TETRIX™ build





**SERVO EXTENSION WIRE**  
Ideal when extra length is needed between the servo and servo controller

**120-TOOTH GEAR**  
Largest TETRIX™ Gear used for gear trains and mechanical mechanisms

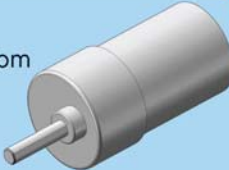



**SERVO Y ADAPTOR**  
For connecting two servos to one and ensuring both are in sync

**80-TOOTH GEAR**  
Midsize TETRIX™ Gear used for gear trains and mechanical mechanisms




**DC DRIVE MOTOR**  
12V motor featuring 152 rpm and 300 oz-in. of torque




**40-TOOTH GEAR**  
Smallest TETRIX™ Gear often used as the pinion gear for gear trains and mechanical mechanisms




**MOTOR MOUNT**  
Holds DC Drive Motor securely to structural elements



**BRONZE BUSHING**  
Reduces friction for axles



**MOTOR CONNECTION WIRE**  
Connects the DC Drive Motor to the motor controller




**100 MM SHAFT**  
Precision-ground axle used for connecting wheels and gears



**GEAR HUB SPACER**  
Provides adequate spacing between wheels and gears. Includes four cap screws for each spacer



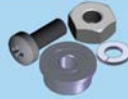
**AXLE SET COLLAR**  
Locks a rotating axle to secure the horizontal alignment





**PIVOT BEARING**

Replacement pivot bearing for the Servo Joint Pivot Bracket

**3/8" NYLON SPACER**

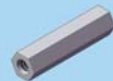
Adds space between wheels, gears, and other elements on axles

**KEP NUT**

Star washer that locks down on structural elements for added strength

**1" STAND-OFF POST**

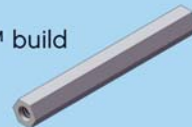
Used as risers in a TETRIX™ build

**3/8" BUTTON HEAD CAP SCREW (BHCS)**

6-32 screw with a low-profile head; used where moving parts are close to the screw heads

**2" STAND-OFF POST**

Used as risers in a TETRIX™ build

**1/2" SOCKET HEAD CAP SCREW (SHCS)**

Basic screw for connecting structural elements

**HEX KEY 4-PACK**

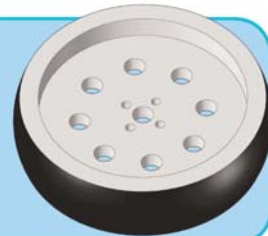
Tools used to fasten screws – four sizes include 7/64", 3/32", 1/16", and 5/64"

**5/16" SOCKET HEAD CAP SCREW (SHCS)**

Basic screw for connecting structural elements

**4" WHEEL**

Large nylon wheel for mobility; features a traction belt

**1/8" NYLON SPACER**

Adds space between wheels, gears, and other elements on axles

**3" WHEEL**

Nylon wheel for mobility; features a traction belt



**OMNI-WHEEL**  
 Multidirectional wheel used for straight and side-to-side movement



**LEGO® HARD-POINT CONNECTOR**  
 Unique piece that connects LEGO® Technic beams to TETRIX™ components



**RECHARGEABLE BATTERY PACK**  
 10-cell, 12V, NiMh rechargeable battery that supplies power to electrical components



**HITECHNIC DC MOTOR CONTROLLER**  
 Connects TETRIX™ DC Drive Motors to the LEGO® NXT Intelligent Brick




**NIMH CHARGER**  
 Recharges the Rechargeable Battery Pack



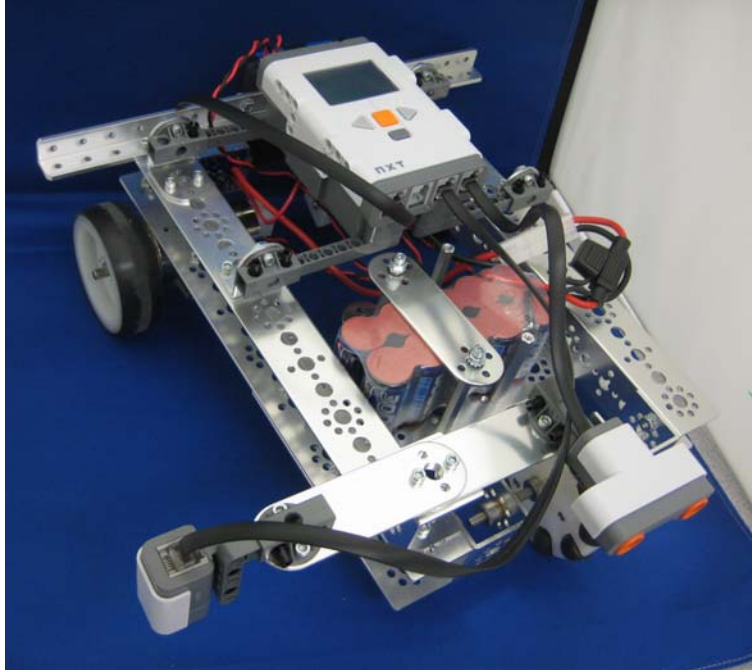
**HITECHNIC SERVO CONTROLLER**  
 Connects TETRIX™ servos to the LEGO® NXT Intelligent Brick



**ON/OFF BATTERY SWITCH**  
 Controls the power from the battery pack



TETRIX rescue vehicle:



Parts needed:

- 2 – 288 mm channels
- 4 – 32 mm channels
- 1 – 288 mm angle
- 4 – L brackets
- 5 – flat brackets
- 1 – 64 x 192 mm flat building plate
- 2 – DC motor mounts
- 4 – 2" stand-off posts
- 4 – 1.5" socket head cap screws (SHCS)
- 1 – 3" omniwheel
- 1 – 100mm shaft
- 3 – bronze bushings
- 4 – axel set collars
- 2 – motor hubs
- 1 – HiTechnic DC motor controller
- 1 – on/off battery switch
- 2 – motor connection wires
- 1 – battery connector wire
- 1 – 12V battery pack
- 5 – LEGO hard point connectors
- 2 – 3" hub/tire
- 45 – 0.375" socket head cap screws (SHCS)
- 45 – kep nuts
- 2 – DC drive motors
- 1 – NXT brick
- 2 – 11-hole technic beam
- 2 – 9-hole technic beam
- 3 – NXT cables
- 13 – friction pins

**Step 1: Wheel-Motor Assembly (x2)**

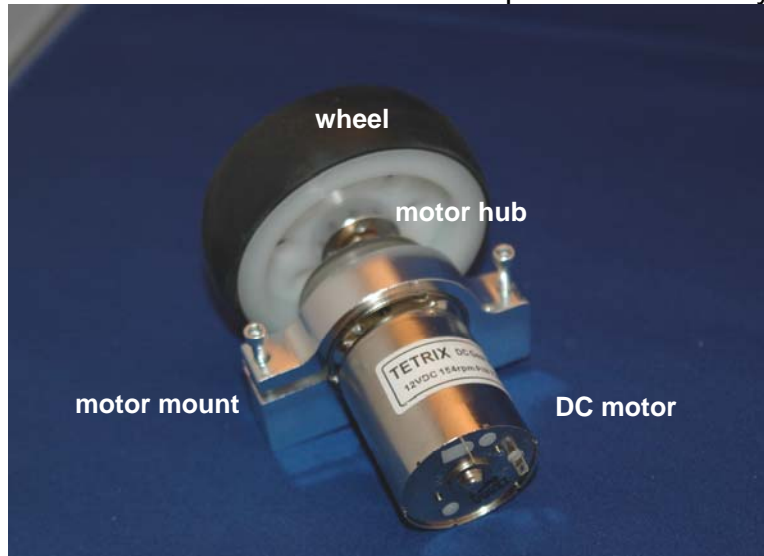
- a. Insert the DC motor into the motor mount.



- b. Put the motor hub on the motor shaft. Make sure the setscrew is positioned over the flat edge of the shaft before you tighten the setscrew.

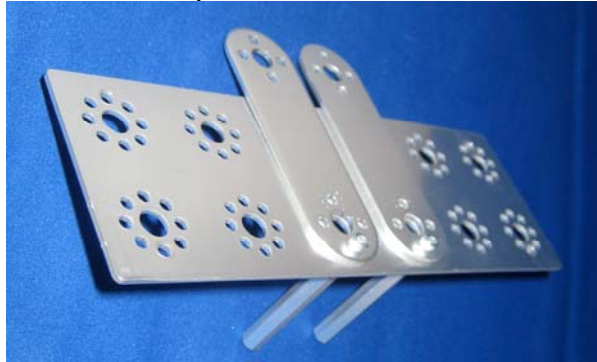


- c. Screw the wheel onto the motor hub to complete the assembly.

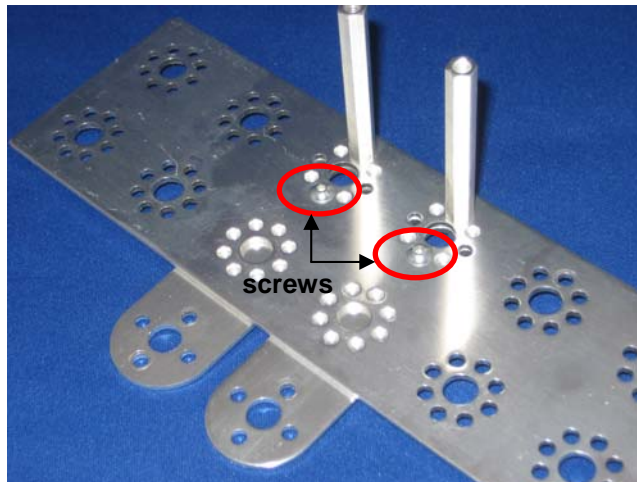


**Step 2: Battery Holder (x1)**

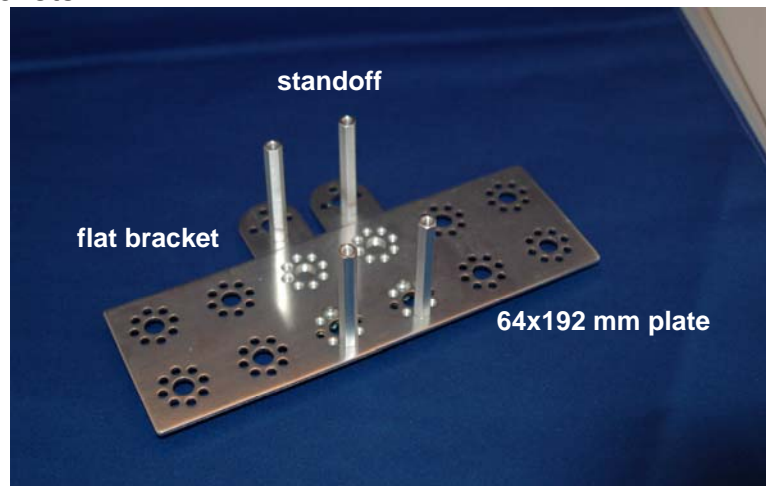
- a. Attach two flat brackets to the 64x192mm plate with the screws coming from the bottom up into the standoffs.



- b. Make the structure more stable with two additional screws coming from the top.

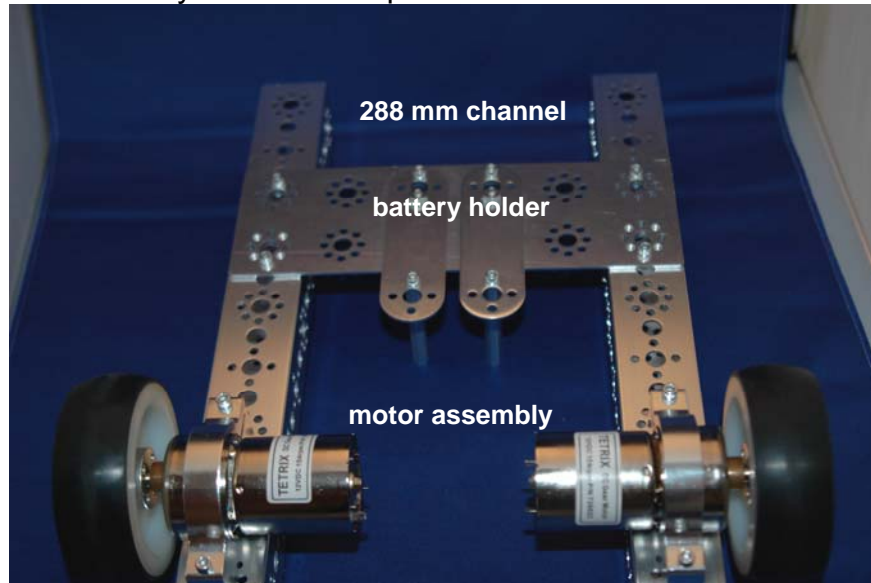


- c. Use two more screws to attach the two remaining standoffs to the flat brackets.



**Step 3: Base (x1)**

- a. Take two 288 mm channels and lay them down on their sides with the open part facing outward.
- b. Attach each of the motors as shown below, using the 1.5" screws and kep nuts.
- c. Attach the battery holder in the position shown below.

**Step 4: Front Wheel Assembly (x1)**

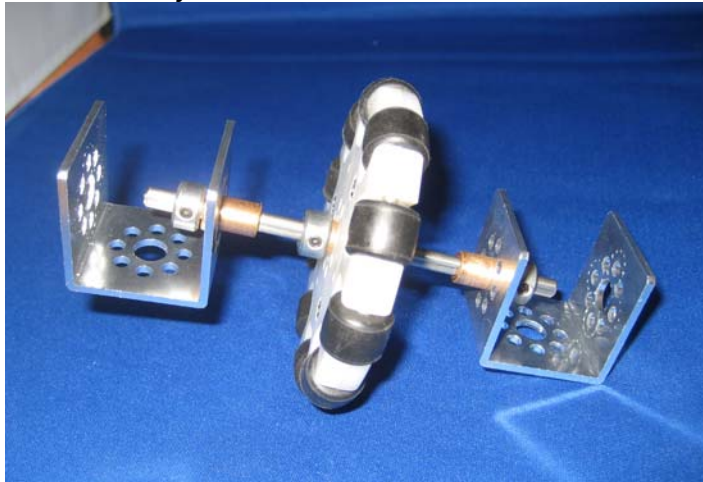
- a. Put a bronze bushing on the 100 mm shaft. Then put the omniwheel over the bushing and place an axel set collar on each side.



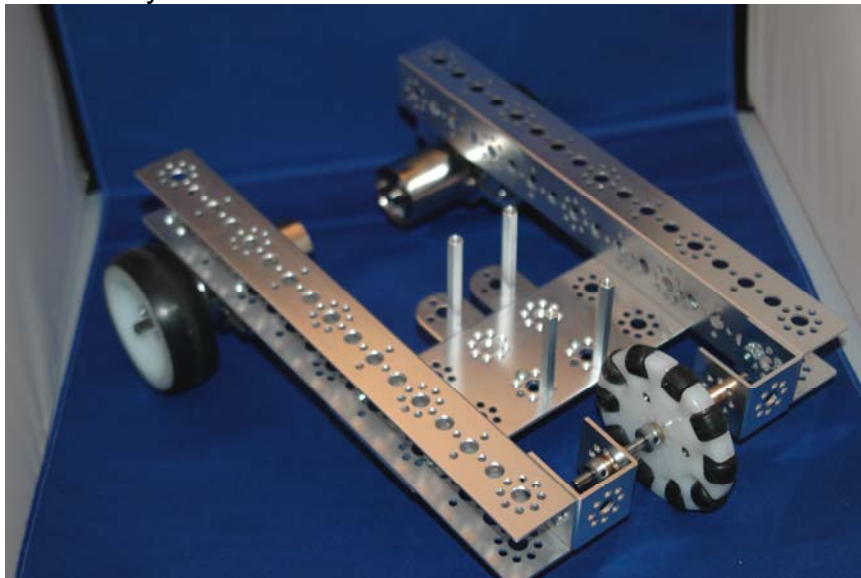
- b. Put a bushing through the large hole on the side of the 32 mm channel.



- c. Slide the 100mm shaft into each of the bushings. Lock with an axel set collar on each side. Adjust the collars to center the omniwheel.

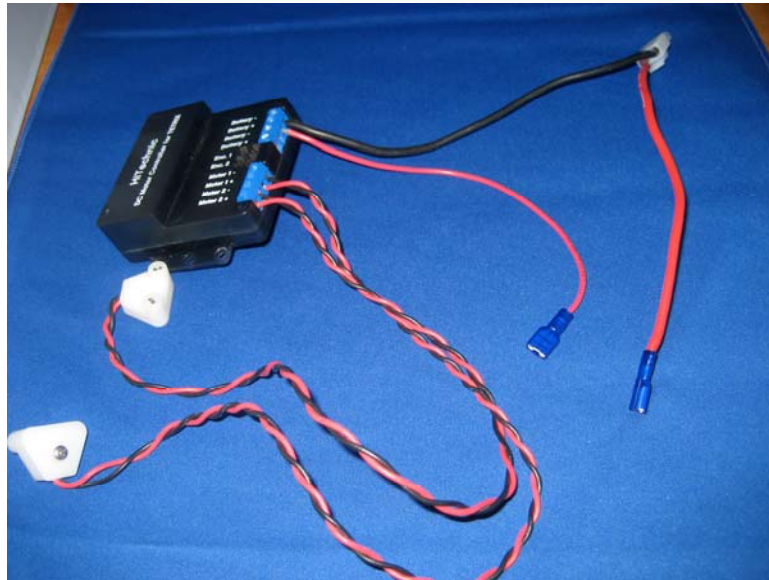


- d. Use two screws-kep nut combinations on each side to attach the front wheel assembly to the base.



**Step 5: Control Assembly (x1)**

- a. Connect the unfinished end of the motor connection wire for the *left* motor (as viewed from the back of the vehicle) to “Motor 1” terminal on the DC motor controller. Connect the black wire to the negative terminal and the red wire to the positive terminal.
- b. Connect the unfinished end of the motor connection wire for the *right* motor to “Motor 2” terminal on the DC motor controller.
- c. Connect the unfinished end of the battery connector wire to the “Battery” terminal on the DC motor controller.



- d. Connect the on/off switch to the finished ends of the battery connector wire.

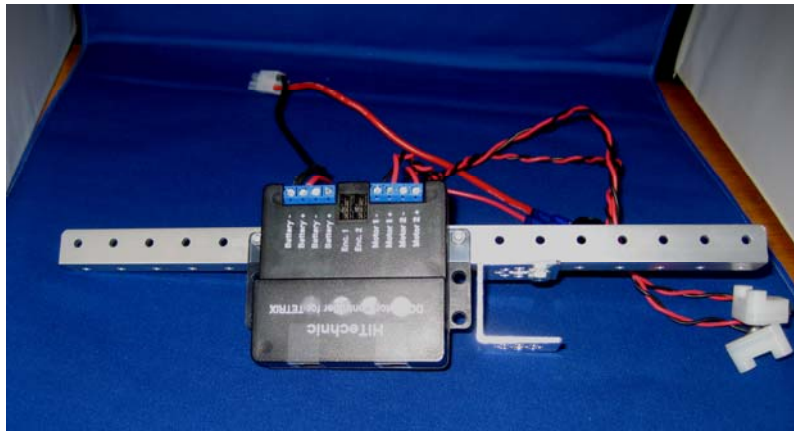




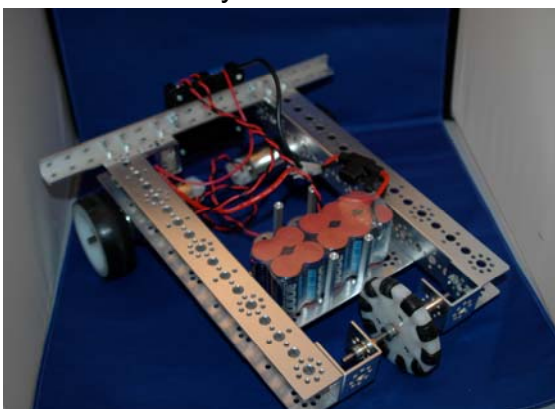
- e. Connect the 288 mm angle bar to the 32 mm channel. Use only one screw-kept nut combination, so that the switch will be able to fit in the channel.



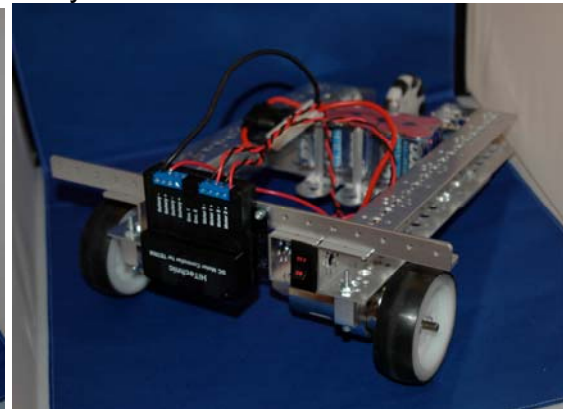
- f. Attach the DC motor controller to the 288 mm angle bar. Don't tighten the screws too much or the plastic housing on the controller will crack. (The battery will be attached once it is positioned in the battery holder in a later step.)



- g. Use two screw-kept nut combinations on either side to attach the control assembly as shown. Place the battery in the battery holder and connect the battery wire to the control assembly.



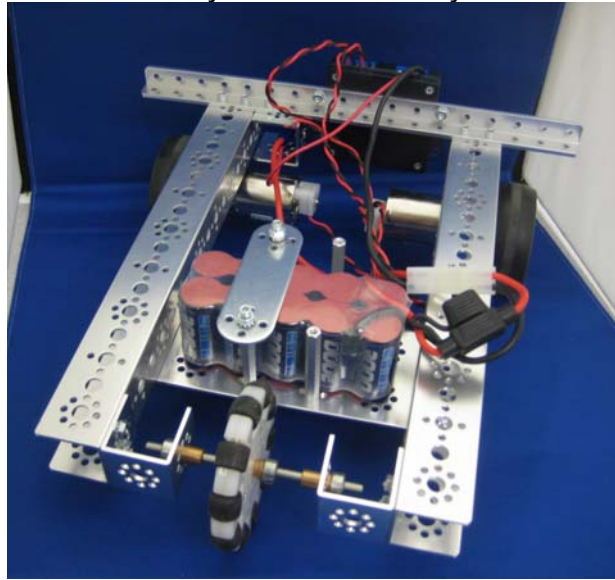
Front view



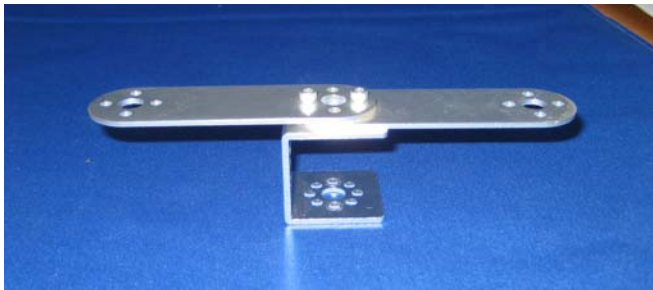
Back view

**Step 6: Battery Lock**

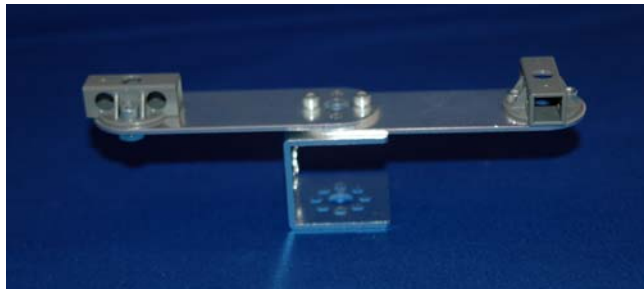
Use two screw-kep nut combinations to attach a flat bracket to the top of standoffs that are part of the battery holder assembly.

**Step 7: Sensor Attachment Assembly (x1)**

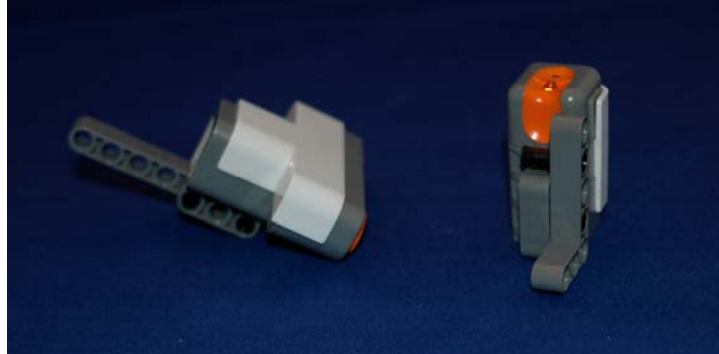
- a. Place a 32 mm channel on its side. Use two screw-kep nut combinations to attach the two flat brackets, in opposite directions, to the 32 mm channel.



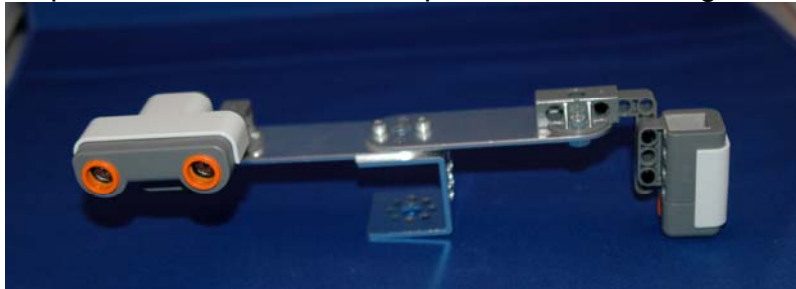
- b. Attach two LEGO hard point connectors with two screw-kept nut combinations on either end of the flat brackets.



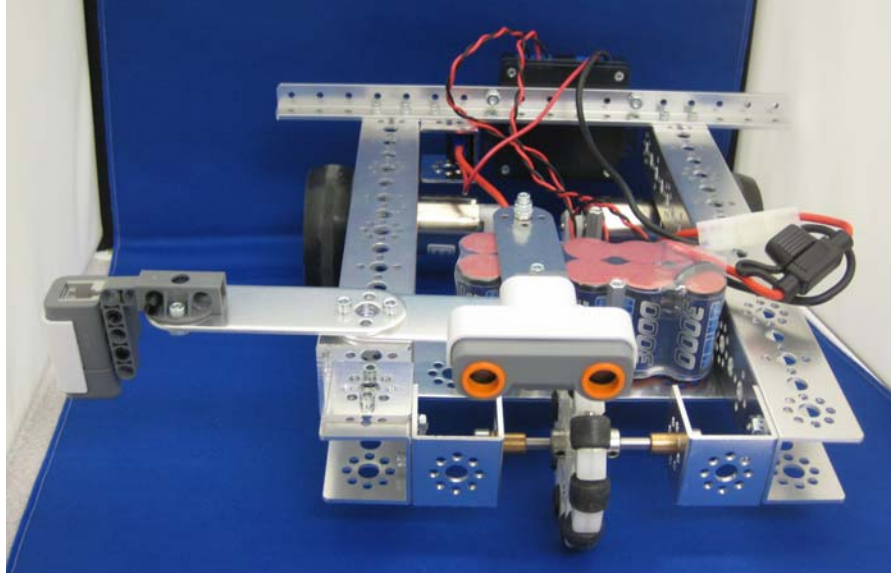
- c. Connect a 7-hole technic beam to the left side of the ultrasonic sensor with two friction pins. Connect a 3x5 L-beam to the light sensor with two friction pins.



- d. Use two friction pins to attach the ultrasonic sensor to the LEGO hard connector piece and use one friction pin to connect the light sensor.

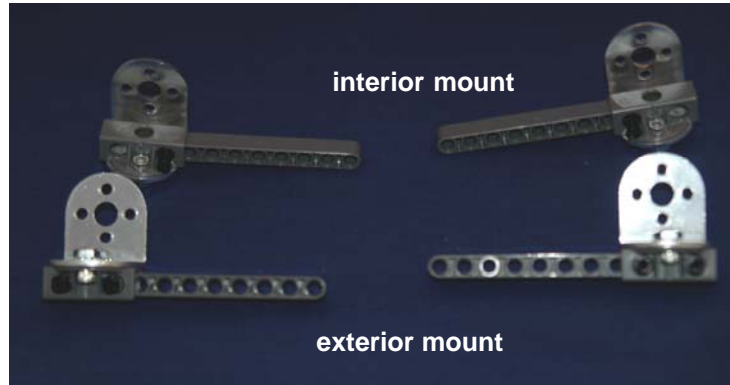


- e. Attach the sensor mounts to the chassis using screw-kep nut combination.



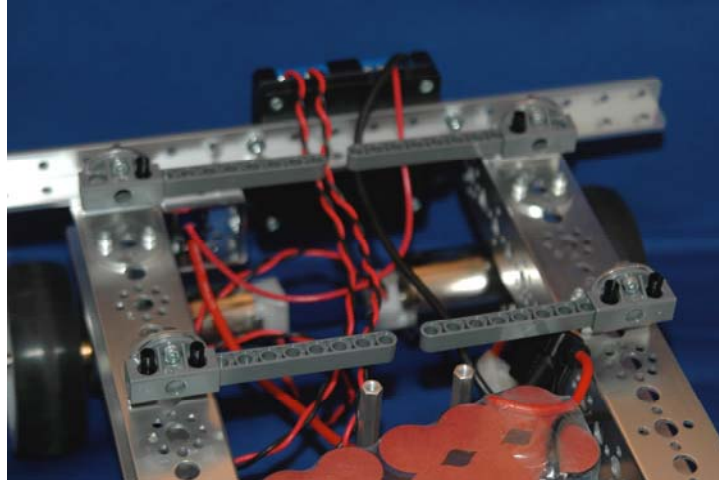
**Step 8: NXT Holder (x1)**

- a. Take two L-brackets and attach LEGO hard connector pieces to the *interior* of the piece. Attach a 9-hole technic beam to these pieces with one pin as shown.
- b. Take two more L-brackets and attach LEGO hard connector pieces to the *exterior*. Attach an 11-hole technic beam to these pieces with two pins, as shown.



Note: There are two different kinds of set-ups shown in the above picture. Two of each set-up will be needed to mount the NXT brick.

- c. Attach the NXT holder to the chassis using screw-kept nut combination.

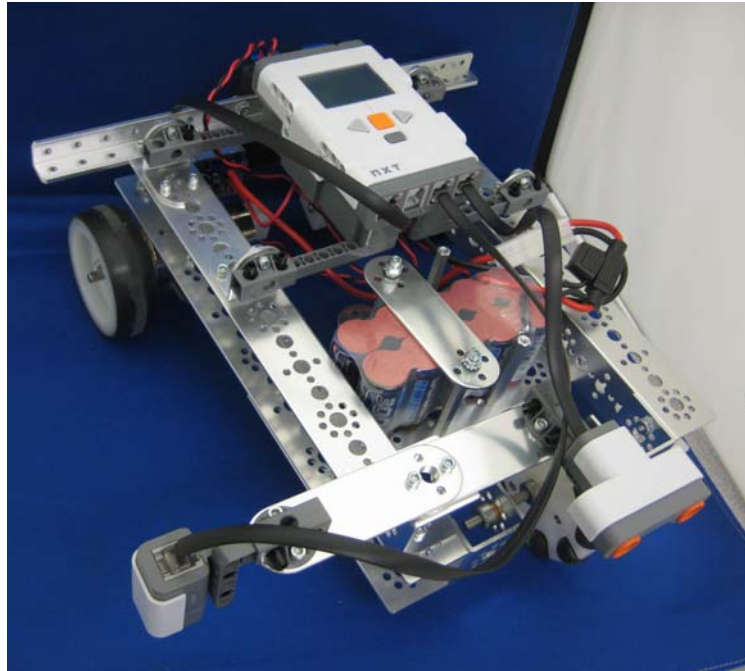


**Step 9: Final Assembly**

- a. Connect NXT cables to the desired ports on the NXT brick.




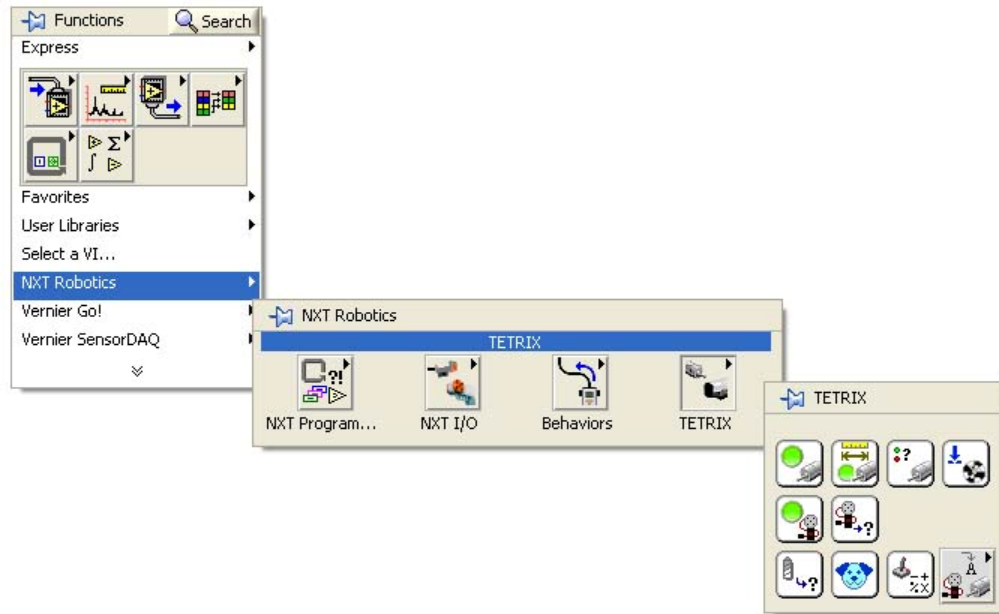
- b. Use four friction pins to attach the NXT brick to the beams on the holder.
- c. Connect the NXT wires to the sensors. Now the robot is finished!



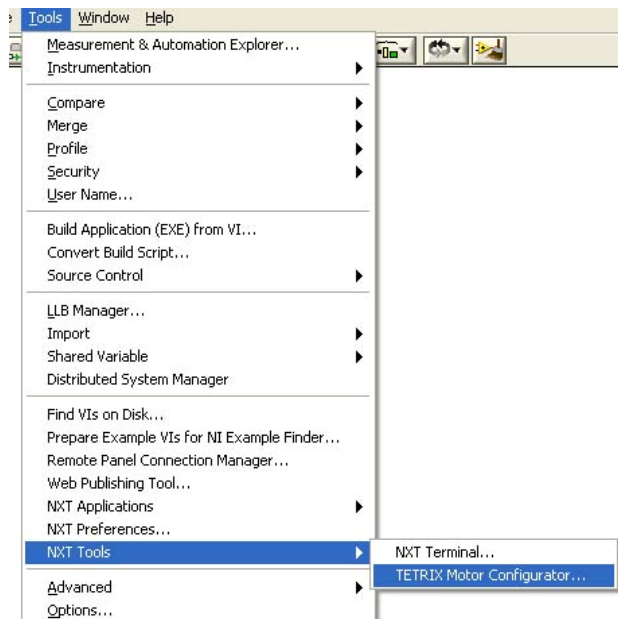
Note: The NXT wire connection in this picture does not match the

### Motor Configurator

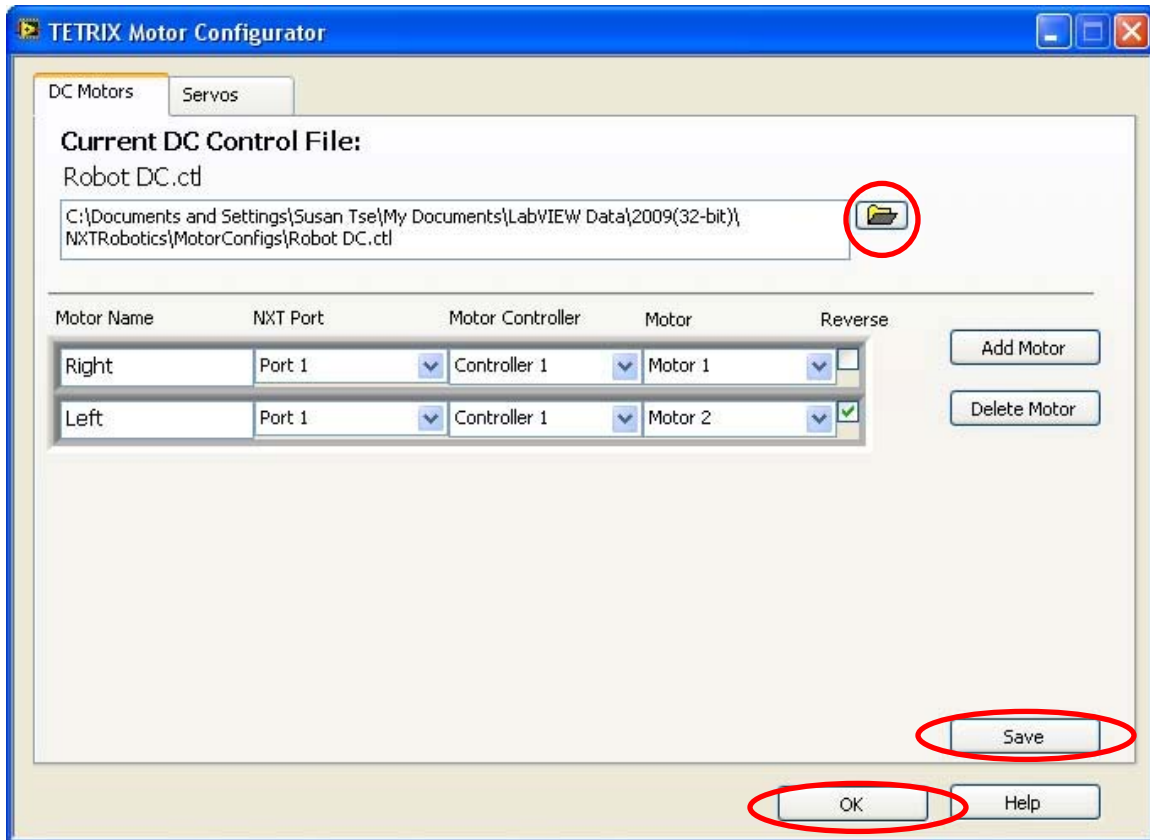
To program the NXT for use with TETRIX, you will also need to use the HiTechnic palette, which you can open by clicking on the  icon.



Before creating a program that will control the TETRIX DC and servo motors, you must open the Motor Configurator from Tools >> NXT Tools >> TETRIX Motor Configurator.

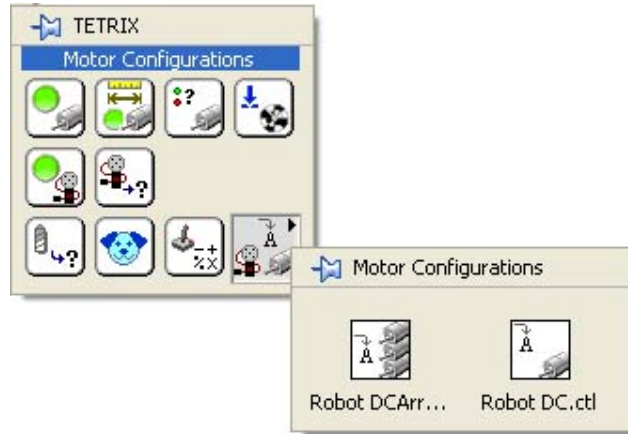


The following window will appear, allowing you to customize your program to your TETRIX vehicle.



The first thing you should do is click on the yellow folder icon to open an existing motor configuration (to modify) or to create a new motor configuration. Next set up the motor configuration, make sure that your TETRIX vehicle set-up matches what you define in the software (i.e. port 4, controller 1, etc.). The motor name is the name you see in the drop-down menu once you put the motor control down in your code. The motor control name is the name you see in the Motor Configurations palette (see below). Click “Save” and “OK” after each control creation.

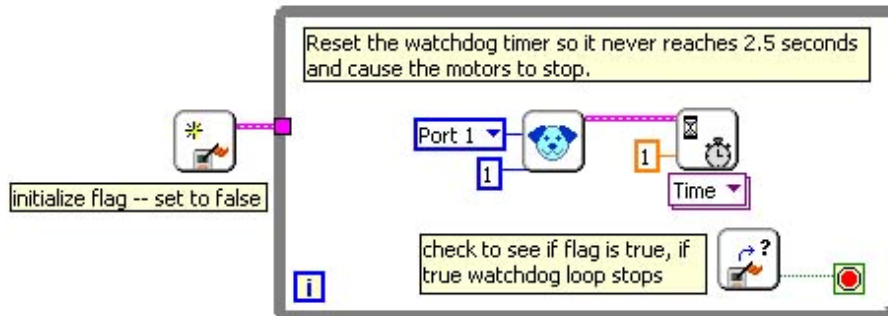
The motor control files created here will be stored in the file directory as shown in the TETRIX Motor Configurator window above, and will be available within LabVIEW from the HiTechnic submenu of the Functions palette.



Note: Make sure that the motor configuration name has an extension of .ctl.

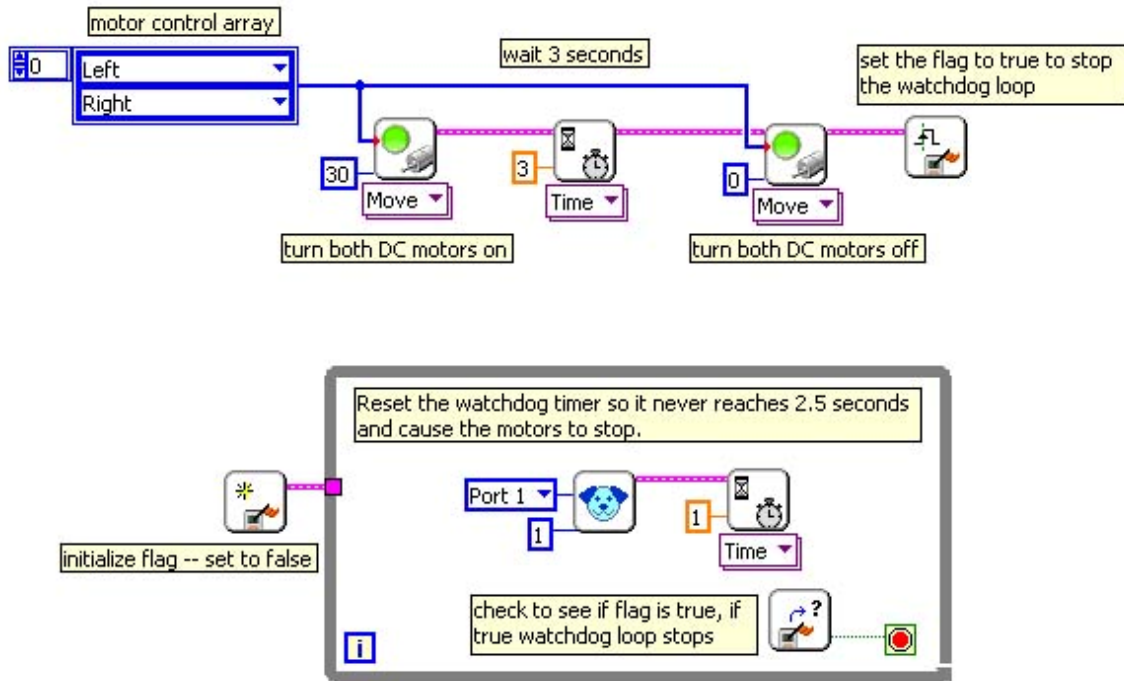
**Watchdog**

The watchdog timer is a safety feature that stops the HiTechnic DC motors after 2.5 seconds. This is to prevent the powerful motors from causing too much inadvertent damage. However for some programs you may want the DC motors to run for more than 2.5 seconds. To keep the motors on for a longer period of time, “feed” the watchdog timer– reset the timer before it reaches 2.5 seconds. One way to rest the timer continuously is to put the watchdog in a while loop, as shown below.

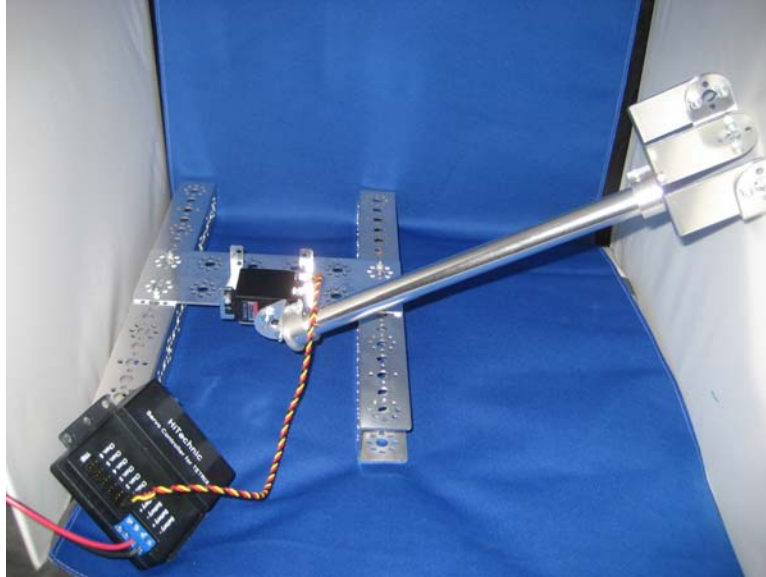




The following program turns the DC motors on for 3 seconds and then stops the DC motors. To prevent the code from running forever, you will need to stop the watchdog while loop by setting the “Stop” control for the watchdog while loop to true (the code in the flat sequence). This will stop the watchdog from being reset.



TETRIX rescue vehicle:



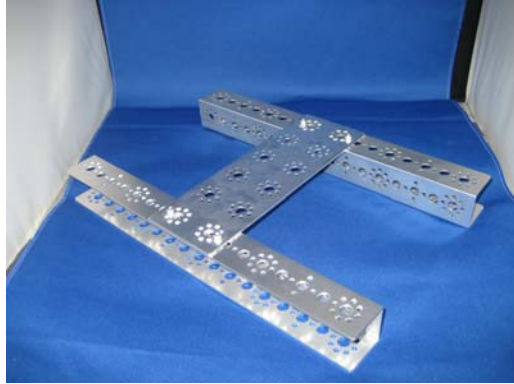
Parts needed:

- 2 – 288 mm channels
- 1 – flat building plate
- 1 – 220 mm tube
- 2 – servo joint pivot bracket
- 1 – double-servo bracket
- 3 – L-bracket
- 20 – 5/16" socket head cap screws (SHCS)
- 16 – kep nuts
- 2 – 1/2" socket head cap screws (SHCS)
- 2 – tube clamp
- 1 – servo motor with horn
- 1 – HiTechnic servo controller

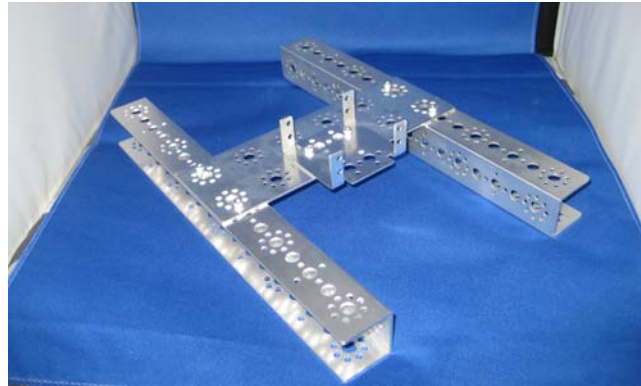
Note: This is a basic arm design. Students should redesign the container design to effectively hold the weighted supplies the teacher has selected.

**Step 1: Base Assembly**

- d. Use two screw-kep nut combinations to secure the flat building plate to the middle of the 288 mm channels.



- e. Attach the double servo bracket to the base using two screw-kep nut combinations.

**Step 2: Arm Assembly**

- d. Use two 5/16" socket head cap screws to attach the tube clamp to the 220 mm tube.



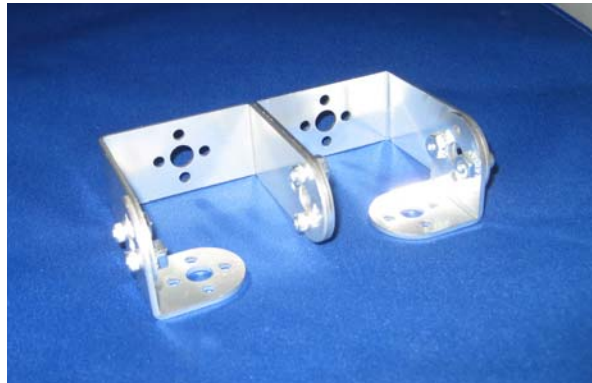
- e. Attach an L-bracket to the servo horn using two screw-kep nut combinations.



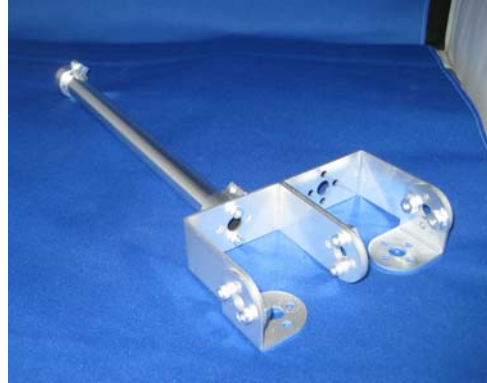
- f. Attach the servo horn to the servo motor. Note: make sure that the L-bracket is horizontal when the servo motor is spun all the way in one direction.



- g. Attach L-brackets to the servo joint pivot brackets using screw-kep nut combinations. Then attach the L-brackets to each other as shown in the picture below using two screw-kep nut combinations. This will be the basic container structure. It should be modified based on the supplies the arm needs to carry.



- h. Attach the 220 mm tube to the container assembly from Step 2d using two screws.



- i. Attach the arm assembly from Step 2e the servo motor assembly (Step 2c) using two screws. Then attach this assembly to the base using 2 screw-kep nut combinations.



**Step 3: Wire-up Servo Motor**

- a. Connect the servo motor wire to the servo controller. Make sure to connect the yellow wire to the first slot (corresponding to the letter Y on the servo controller—see below) on Channel 1.
- b. Power the servo controller by connecting the spare black and red wires in the servo controller pack to the battery terminals of the DC motor controller and the servo controller as show below.



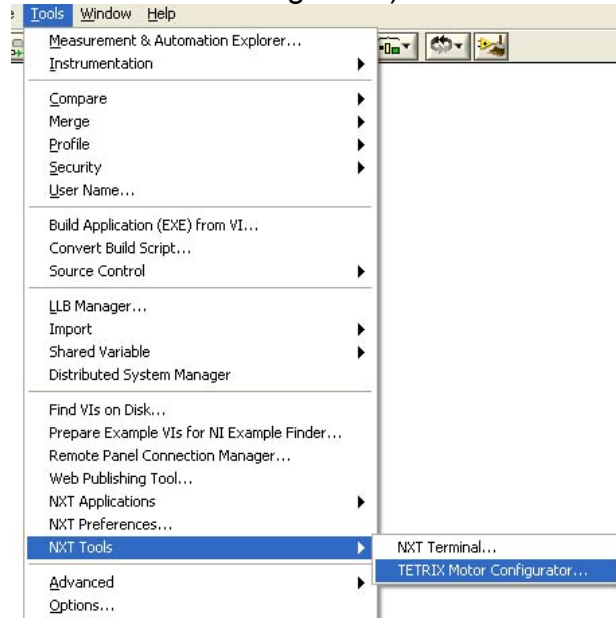
- c. Use an NXT cable to connect the DC motor controller to the servo controller. For this section you can leave the NXT on the TETRIX rescue vehicle.



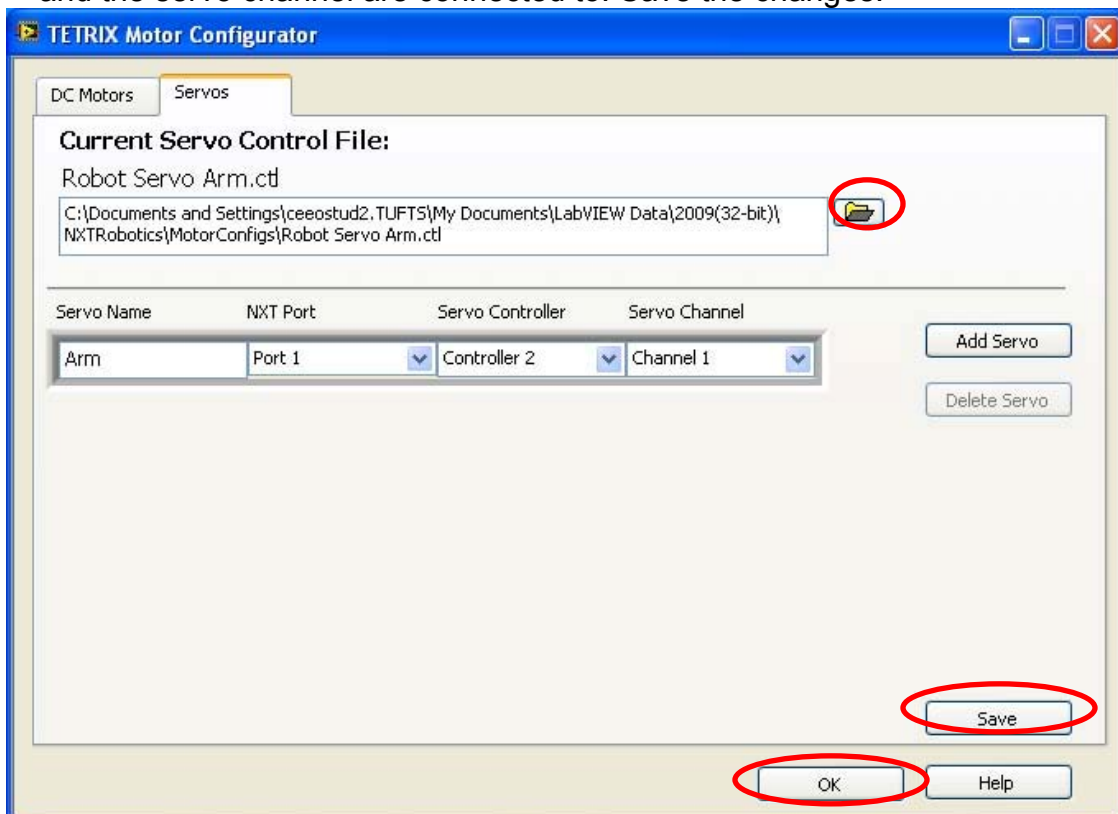
See

[http://www.education.rec.ri.cmu.edu/products/getting\\_started\\_tetrix/labview/testbed/tetrix\\_testbed.html](http://www.education.rec.ri.cmu.edu/products/getting_started_tetrix/labview/testbed/tetrix_testbed.html) for video instruction on how to wire up the servo motor.

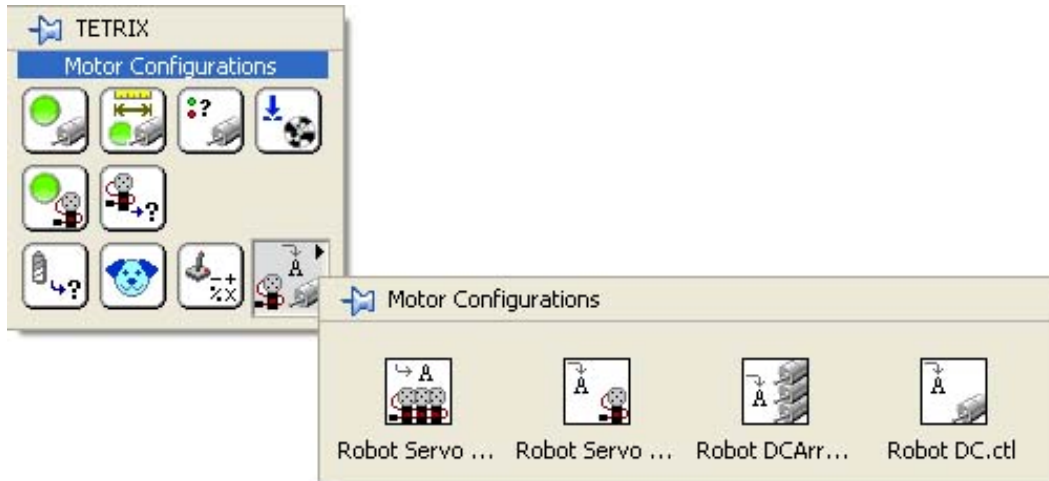
1. Open the Motor Configurator window to create servo motor controls (Tools >> NXT Tools >> TETRIX Motor Configurator).



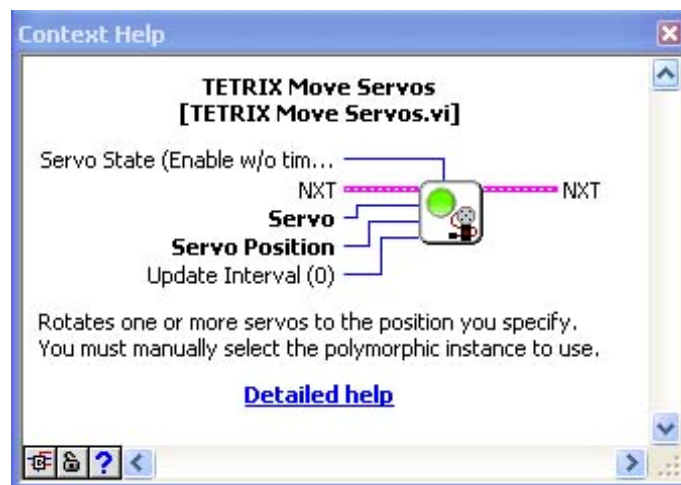
2. Click on the yellow browse folder icon to create a new motor configuration. Then give the servo motor a descriptive name. For example, naming the arm servo motor "Arm". Select the NXT port the controller, the servo controller, and the servo channel are connected to. Save the changes.



- Two new motor controls will be created in the Motor Configuration palette, as shown below. The first one is for an array of servo motors and the second one is for a single servo.

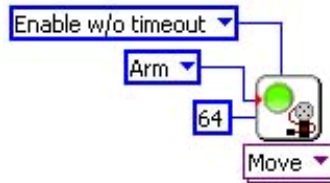


- The context help for the TETRIX Move Servos VI show that there are two required inputs: servo and servo position. The servo input is the motor control that was created using the Motor Configuration window. The servo position input is the position (0-255) of the servo motor. Note that servo position is not measured in degrees—the 180 degrees of motion of the servo maps to 0 to 255, so the servo moves 0.7 degrees per position.

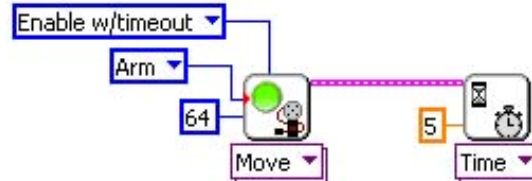




- To program the servo motor to maintain position at a 45 degree angle, make the servo position input 64 (note: position has to be an integer).

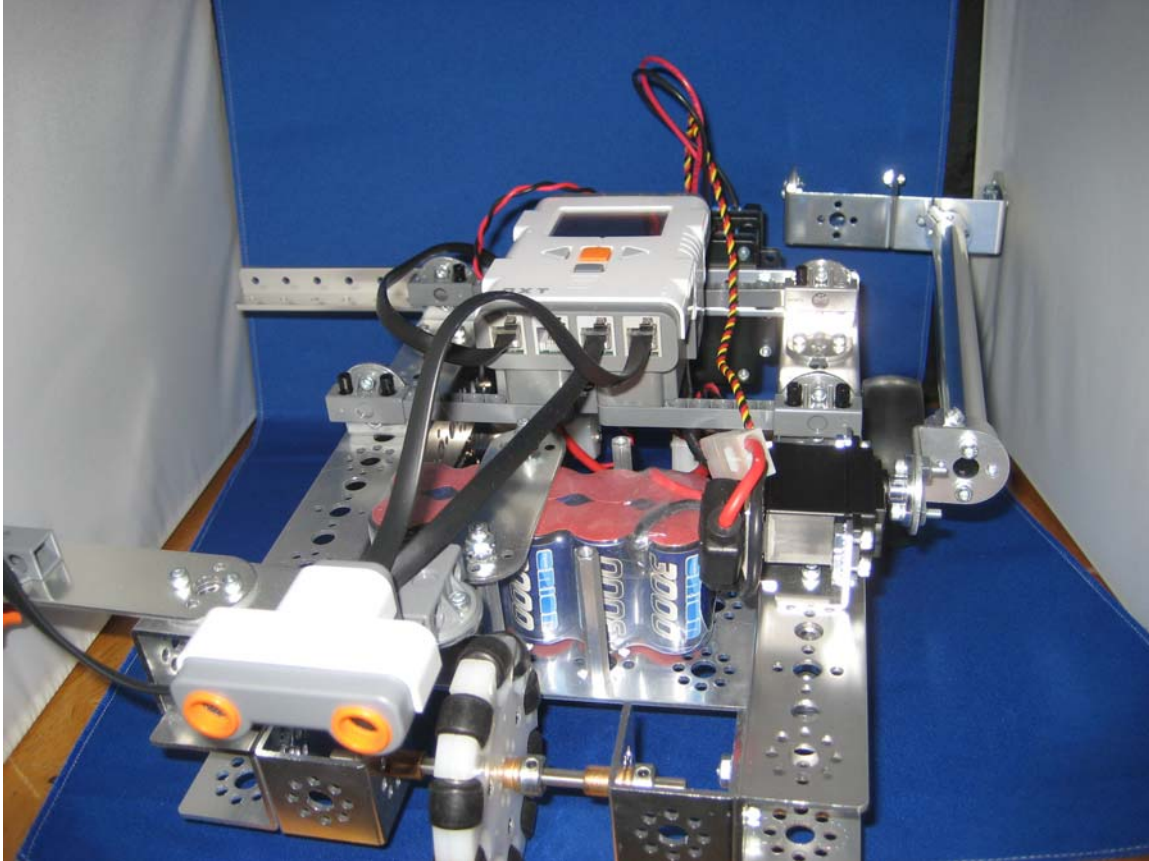


If the servo state input is set to enable without time as shown above, the servo motor will continue to hold the position even after the program ends. To turn off the servo motor, flip the power switch for the 12V battery pack to the off position.



To programmatically stop the servo motor, set the servo state input to enable with timeout and set a time for when the servo motor will stop as shown above.

TETRIX rescue robot:



Parts needed:

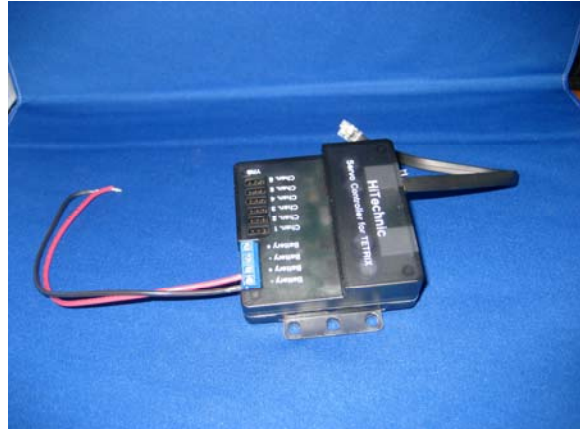
- 1 – TETRIX car assembly (from Lesson 3)
- 1 – TETRIX arm assembly (from Lesson 4)
- 1 – 220 mm tube
- 1 – HiTechnic servo controller
- 1 – Servo bracket
- 6 – 5/16” socket head cap screws (SHCS)
- 6 – Kep nuts
- 4 – 1/2” socket head cap screws (SHCS)
- 2 – 1” standoffs

**Step 1: Disassemble Robotic Arm from Base**

Take the arm you built in Lesson 4 off the base. Keep the arm intact, as you will use it in this section. Note: you will also need the servo controller and corresponding wires. If this is already attached to the system, you can leave it attached.



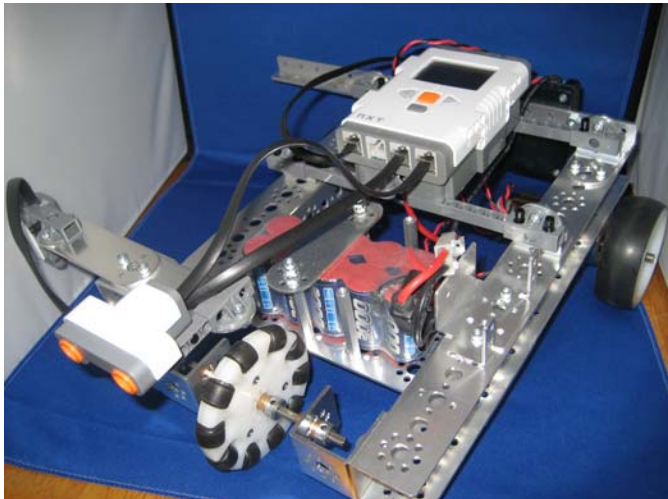
detached arm



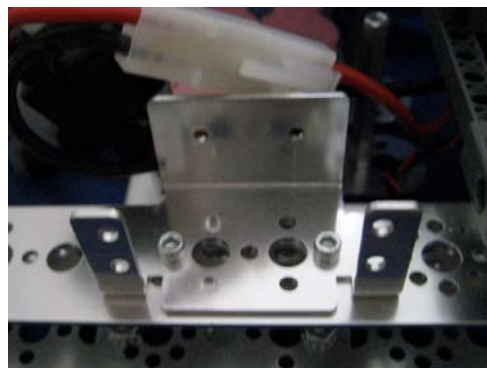
servo controller with wires

**Step 2: Attach the Arm to the Rescue robot**

- j. Use two screw-kep nut combinations to attach the single-servo bracket to the rescue robot chassis as show below.

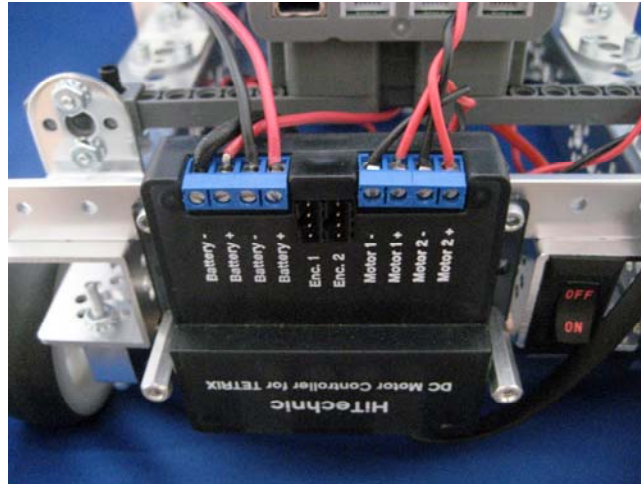


rescue robot chassis with servo bracket

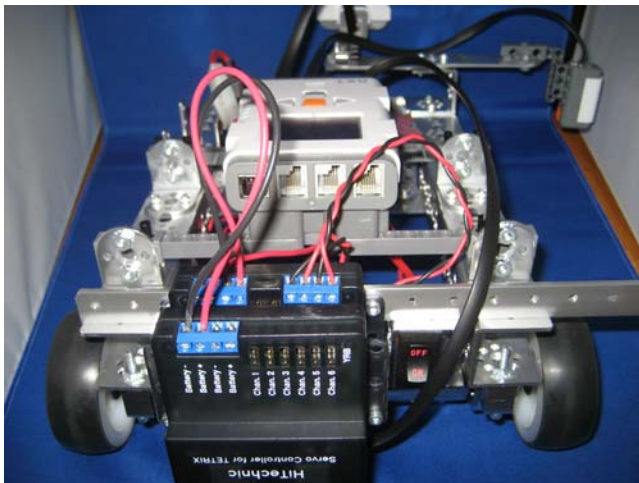


close-up of servo bracket

- k. Attach the 1" standoffs to the DC motor controller as shown below.



- l. Attach the servo controller to the DC motor controller standoffs using two screws.

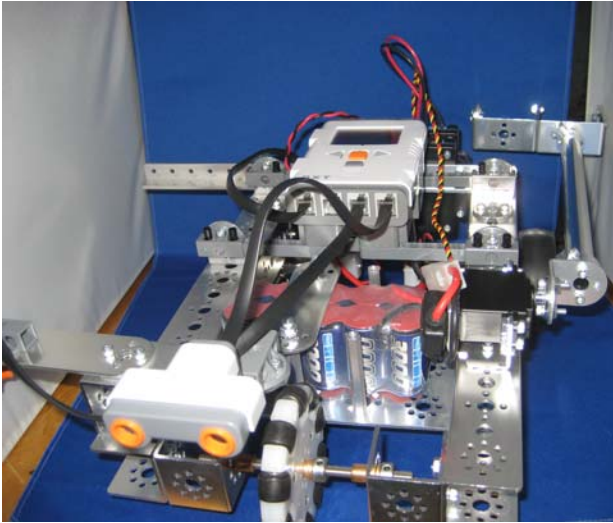


rescue robot chassis with servo controller attached

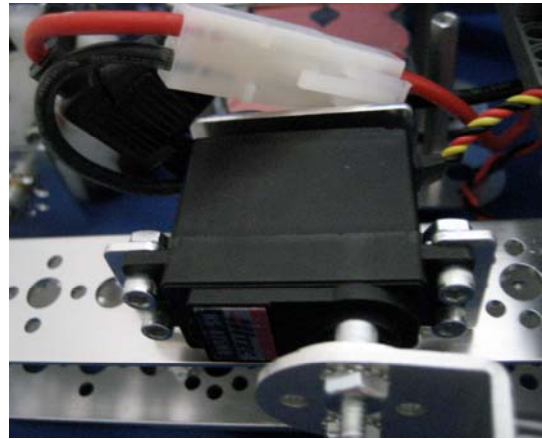


close-up of servo controller attachment

m. Attach the arm to the servo bracket using 4 screw-kep nut combinations.



complete rescue robot with arm



close-up of servo in servo bracket

**Autonomous vehicle:** A vehicle that operates independent of continual input from the user.

**Block diagram:** Part of the LabVIEW program where you write your code by wiring together graphical representation of functions and VIs.

**Boolean:** A logical data type that is either true or false.

**Case Structure:** A programming structure that allows you to choose different behaviors based on the current condition.

**Chassis:** The body of a vehicle includes the main structure, wheels, motors, etc.

**Code:** A series of commands that tells the robot what to do.

**Compile:** Translates the code into a language that the processor can understand.

**DC motor:** Motor that runs on direct current electricity.

**Debugging:** Process of identifying and correcting problems with the code that cause unexpected data or behavior.

**Degrees of freedom (DOF):** Number of independent displacements or rotations in which motion is possible.

**Direct mode:** The NXT brick is connected to the computer via USB or Bluetooth and the program executes on the computer.

**Download:** Loads the compiled program onto the NXT brick.

**Engineering design process:** An eight-step guideline that engineers follow to ensure that their product is designed efficiently and effectively.

**Front panel:** Part of the LabVIEW program where you create a user interface.

**Hypotenuse:** The longest side of a right triangle.

**Light Emitting Diode (LED):** A light emitting source

**Pitch:** For an object heading in the x-direction, pitch is the rotation of the x-axis around the y-axis (e.g. nose up-down motion of a plane flying in the x-direction).

**Potentiometer:** A variable resistor.

**Prototype:** A model of a design on which tests can be performed to evaluate whether the design should be used for the final product.

**Pulse width modulation:** A method of providing varying amounts of electrical power by controlling the duration that power is on and off.

**Pythagorean Theorem:** The square of the hypotenuse of a right triangle equals the sum of the squares of the two shorter sides ( $a^2 + b^2 = c^2$ , where a and b are the short sides a right triangle and c is the hypotenuse).

**Remote mode:** The NXT brick does not need to be connected to the computer and the downloaded program executes on the NXT brick.

**Roll:** For an object heading in the x-direction, roll is the rotation of the y-axis around the z-axis (e.g. spinning about the long axis of the plane).

**Sensor:** A device that robots use to collect information about the environment.

**Servo motor:** A motor that uses mechanical control to maintain a set position.

**Torque:** Force needed to rotate an object about an axis.

**Virtual Instrument (VI):** A LabVIEW program.

**Watchdog:** A programmatic safety mechanism within LabVIEW that turns off the HiTechnic DC motors after 2.5 seconds.

**While loop:** A programming structure that repeats code contained inside the loop until the end condition of the loop is met.

**Yaw:** For an object heading in the x-direction, yaw is the rotation of the x-axis around the z-axis (e.g. nose left-right motion of a plane flying in the x-direction).