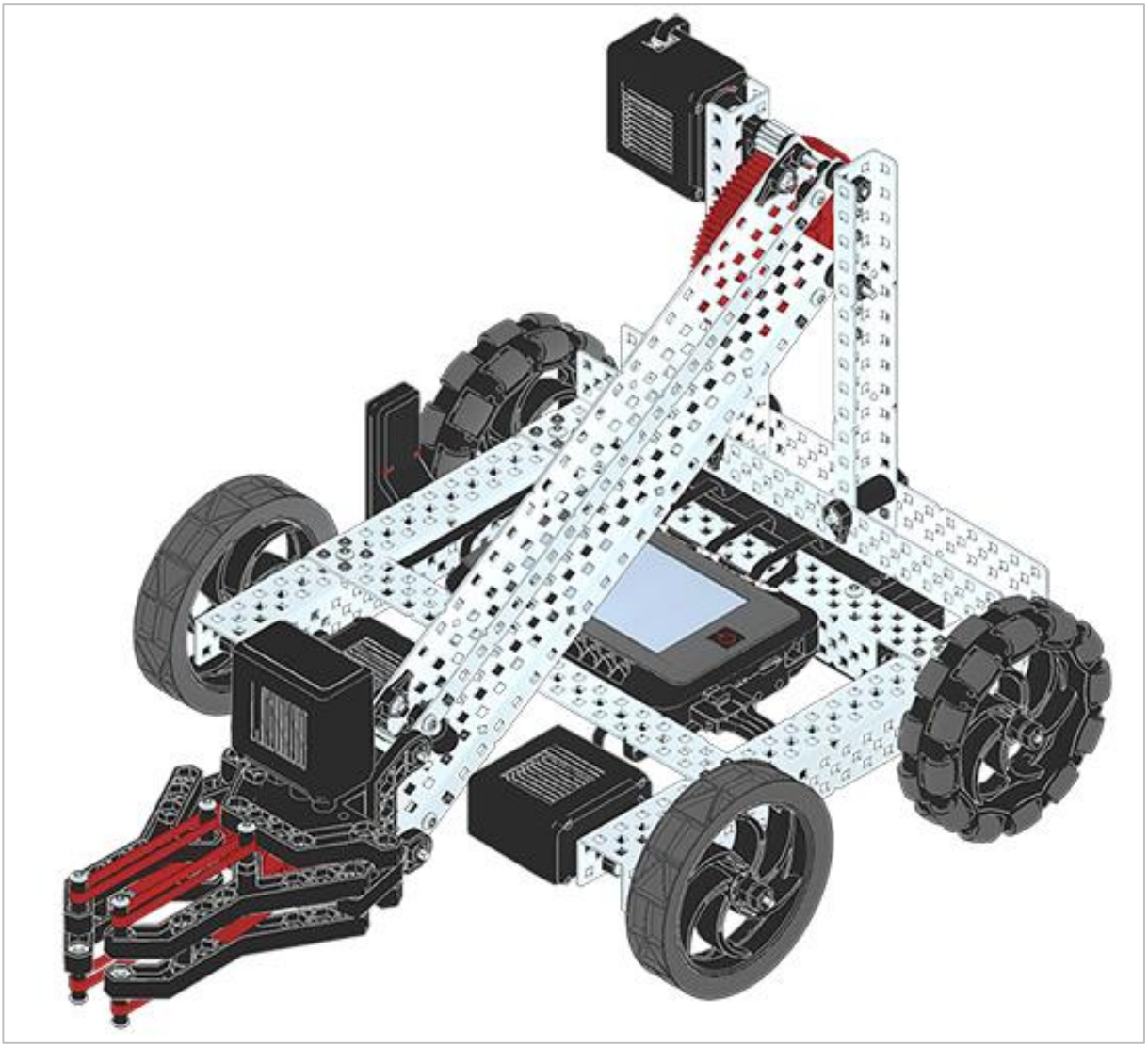
# Vision Sensor

Use the Vision Sensor to detect objects!


Discover new hands-on builds and programming opportunities to further your understanding of a subject matter.

# The Completed Look of the Build



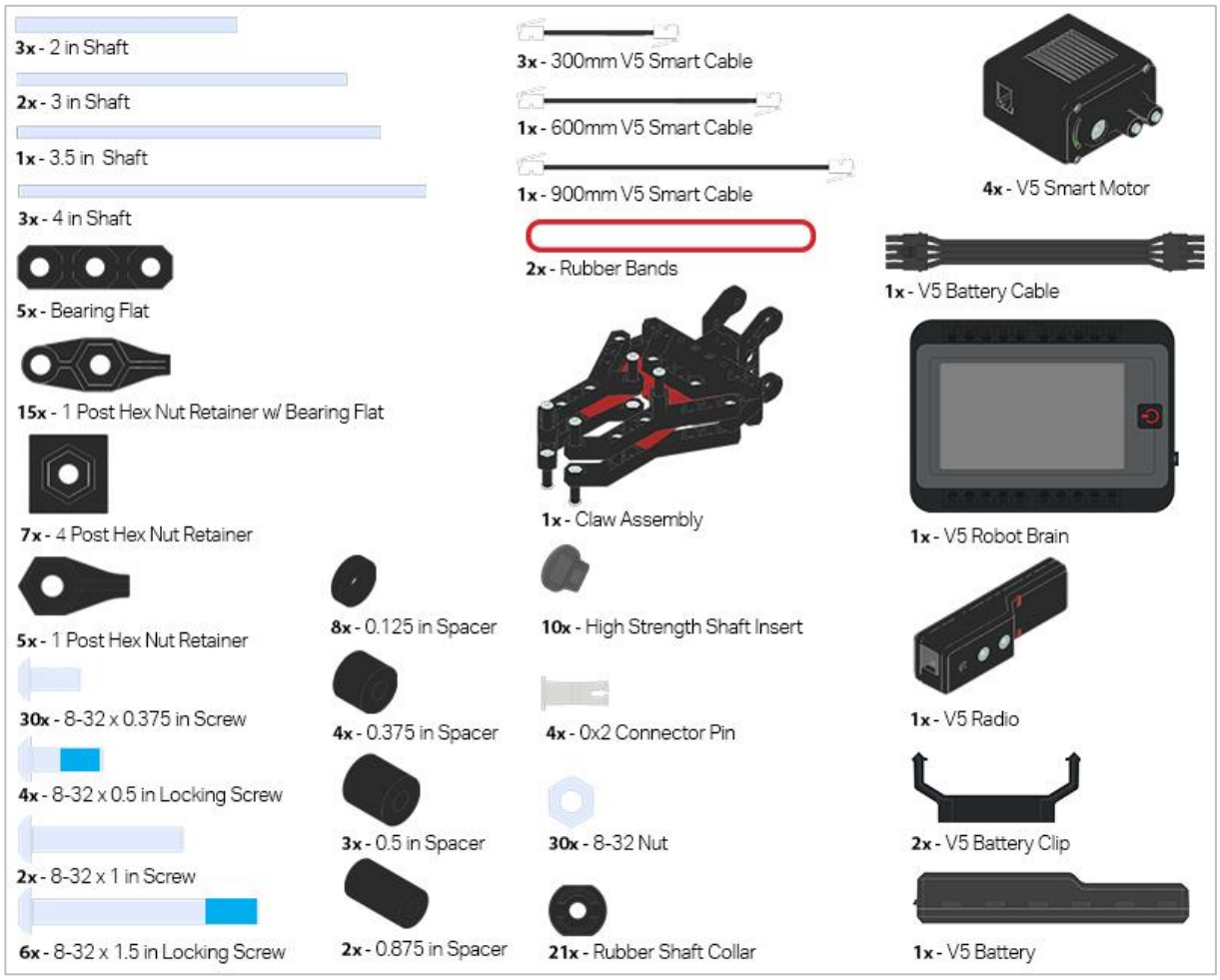
*Completed VEX V5 Clawbot*

The VEX V5 Clawbot is an extension of the VEX V5 Speedbot that can be programmed to move around and interact with objects.

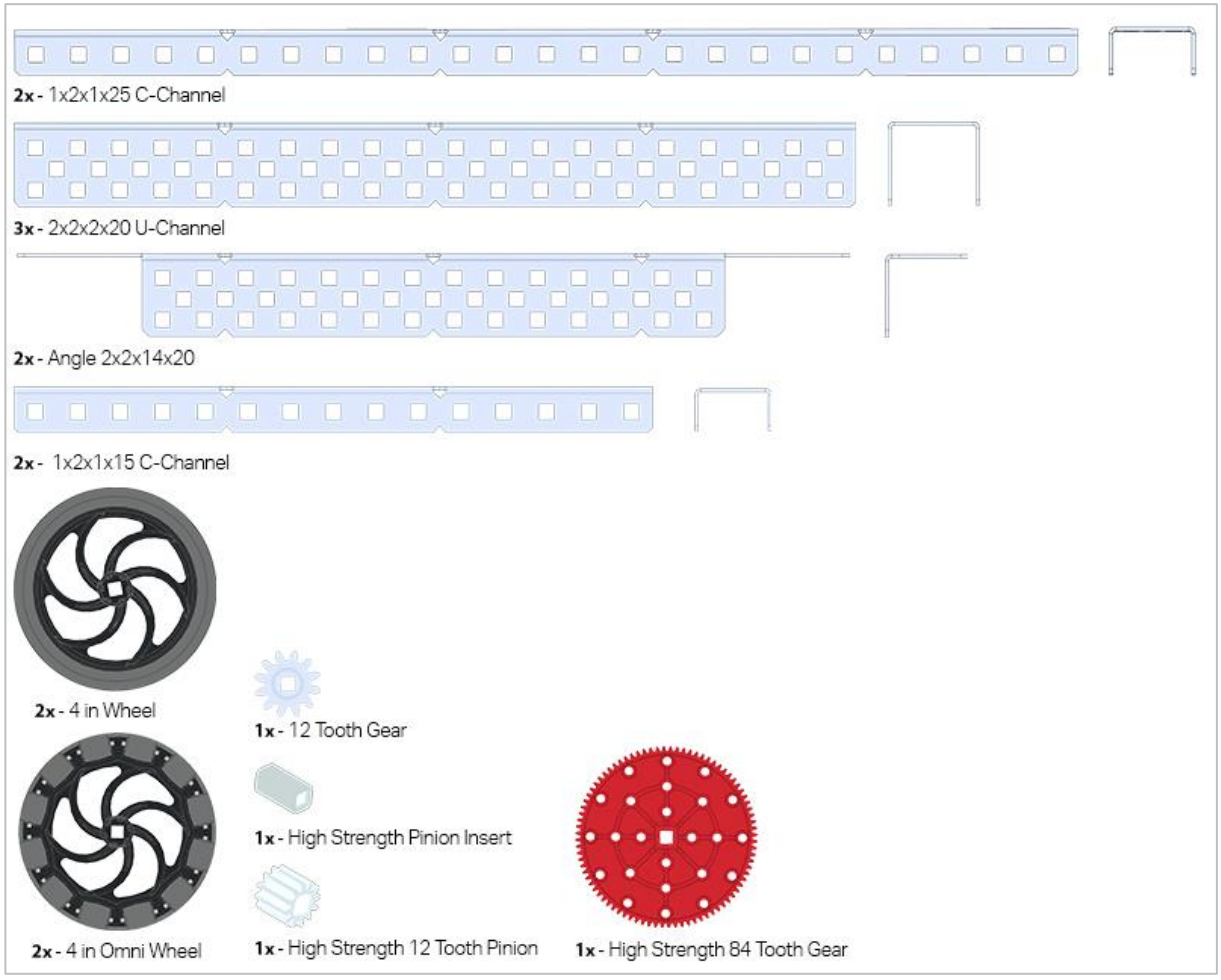
# Parts Needed: Part 1

**Can be built with:**

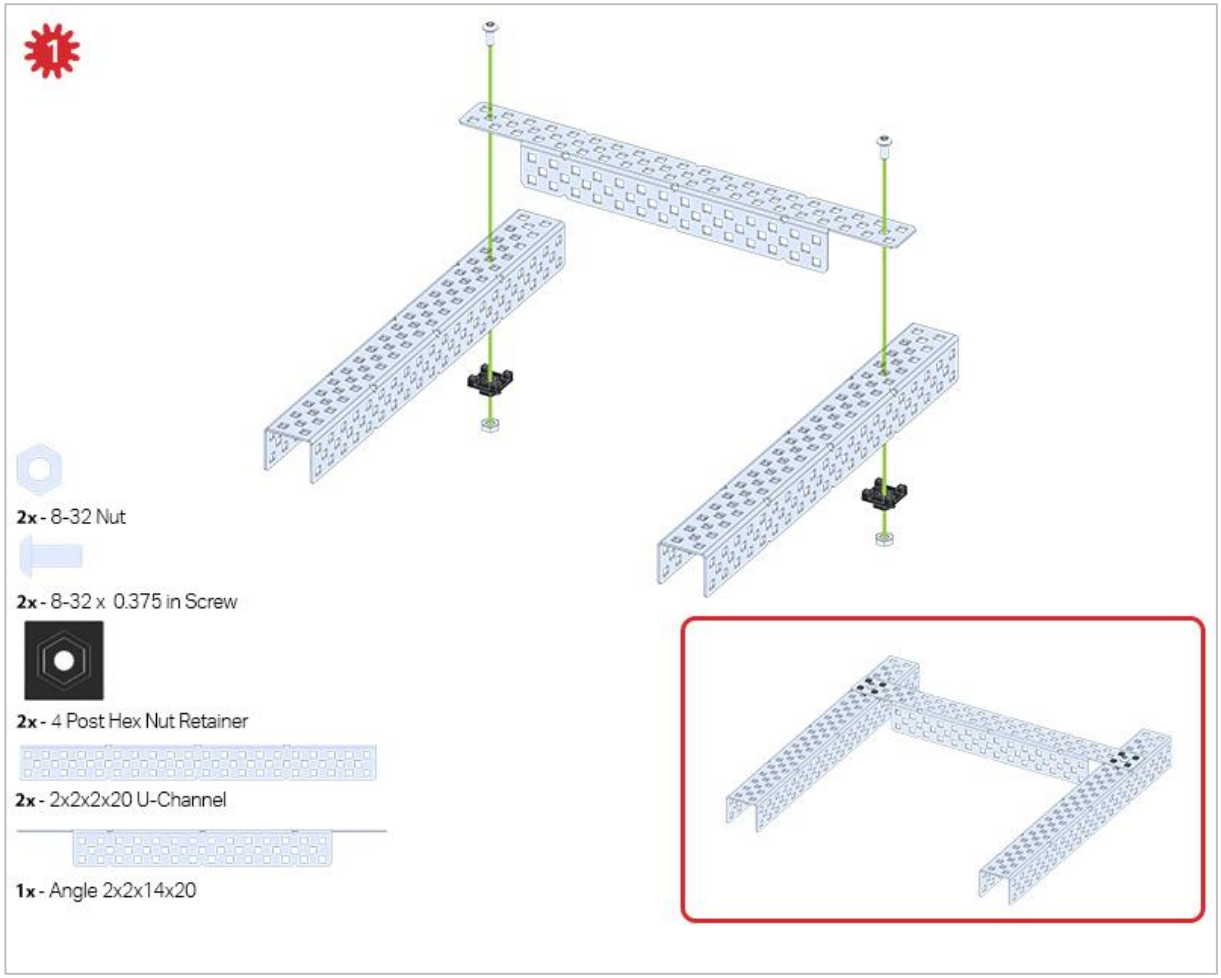
- VEX V5 Classroom Starter Kit



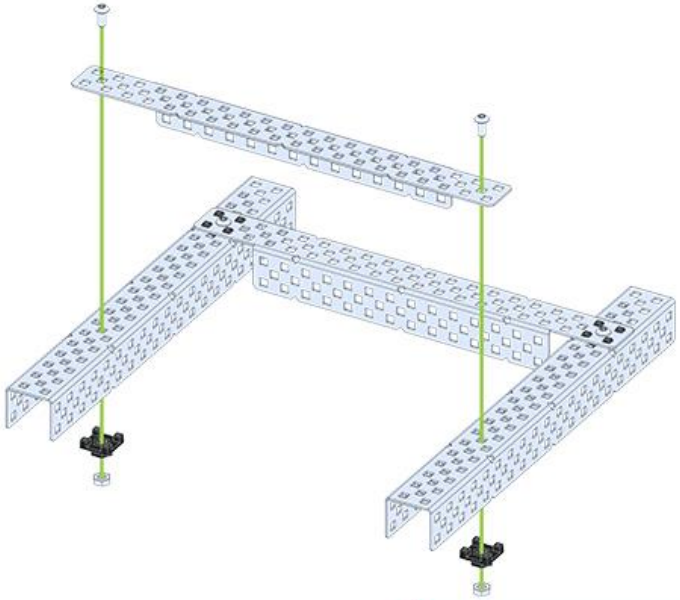
# Parts Needed: Part 2



# Build Instructions



2





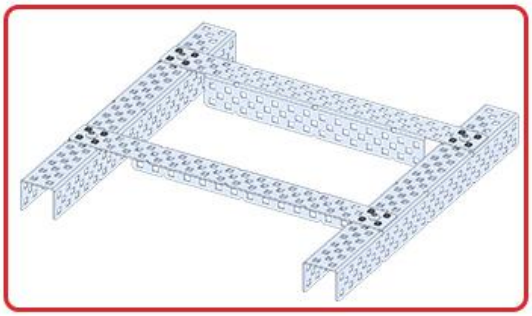
**2x** - 8-32 Nut



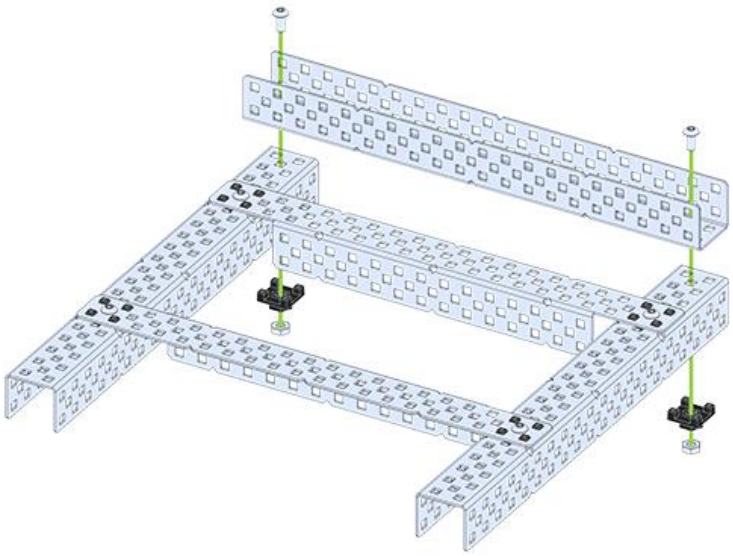
**2x** - 8-32 x 0.375 in Screw

**2x** - 4 Post Hex Nut Retainer

**1x** - Angle 2x2x14x20



3

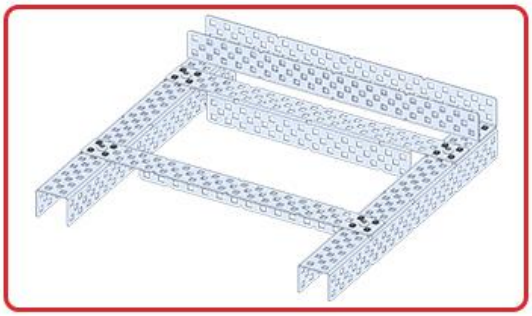


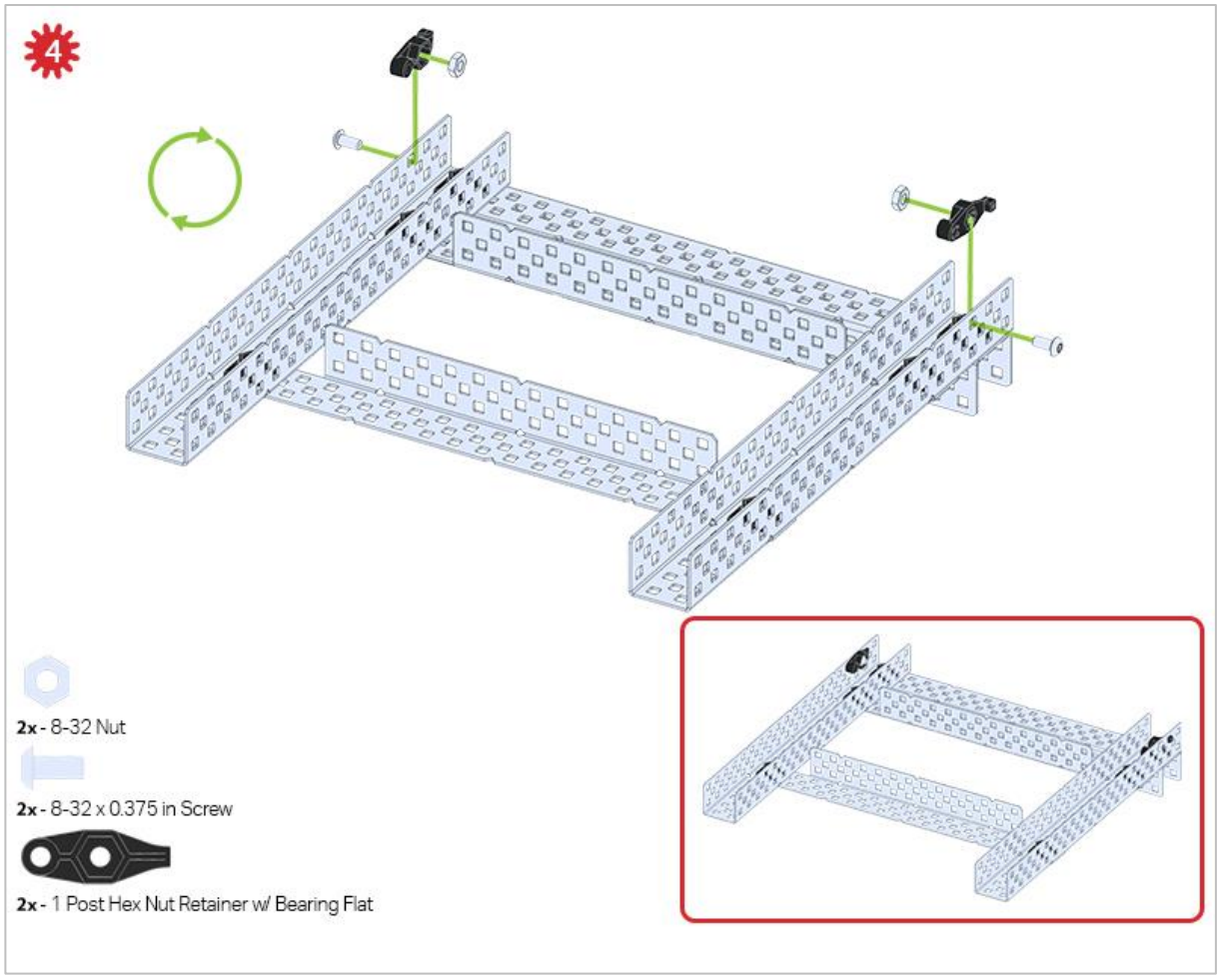
**2x** - 8-32 Nut

**2x** - 8-32 x 0.375 in Screw

**2x** - 4 Post Hex Nut Retainer

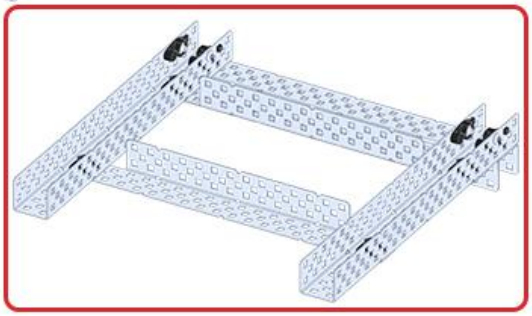
**1x** - 2x2x2x20 U-Channel






*The green icon indicates that the build needs to be flipped over (upside down).*

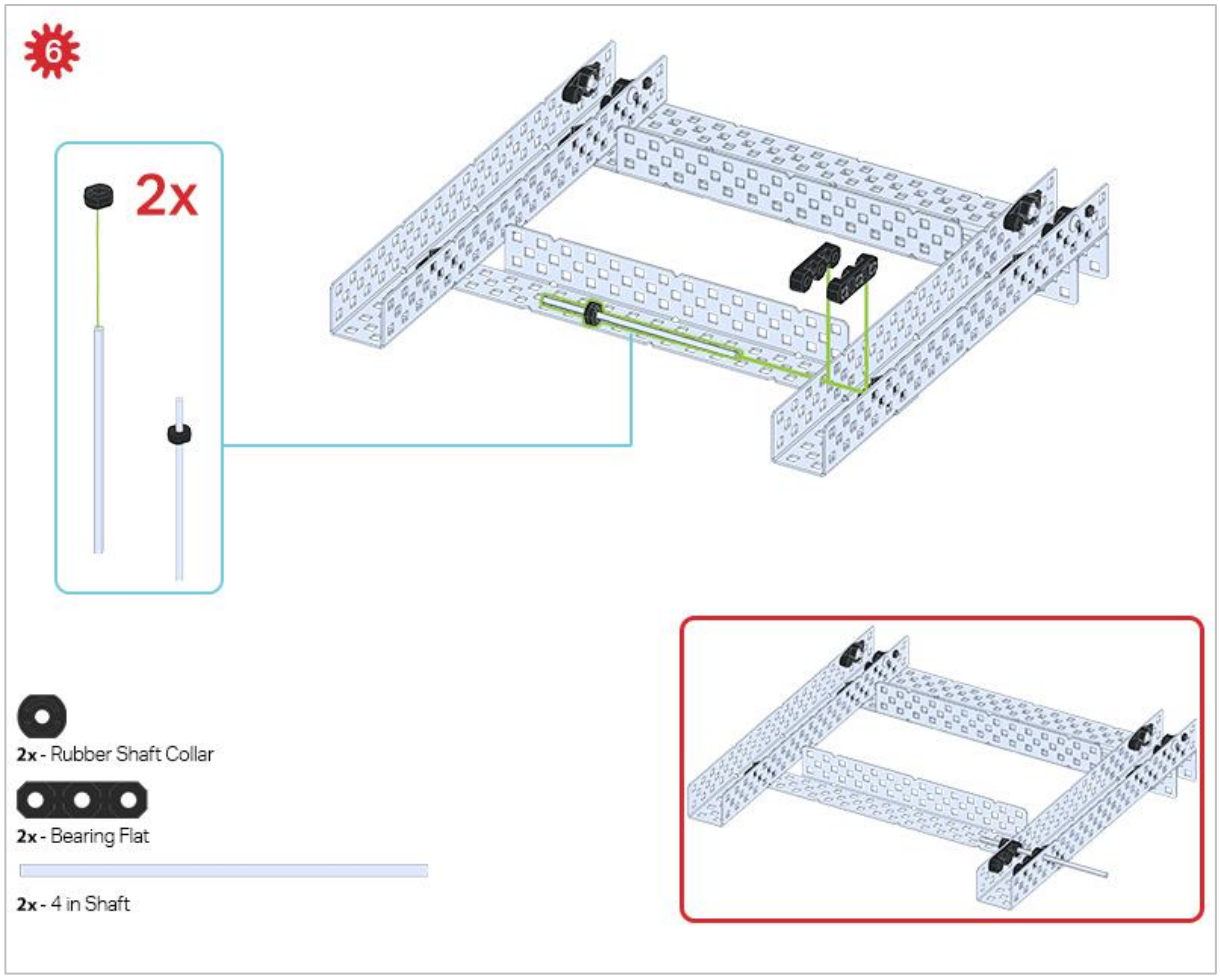
5



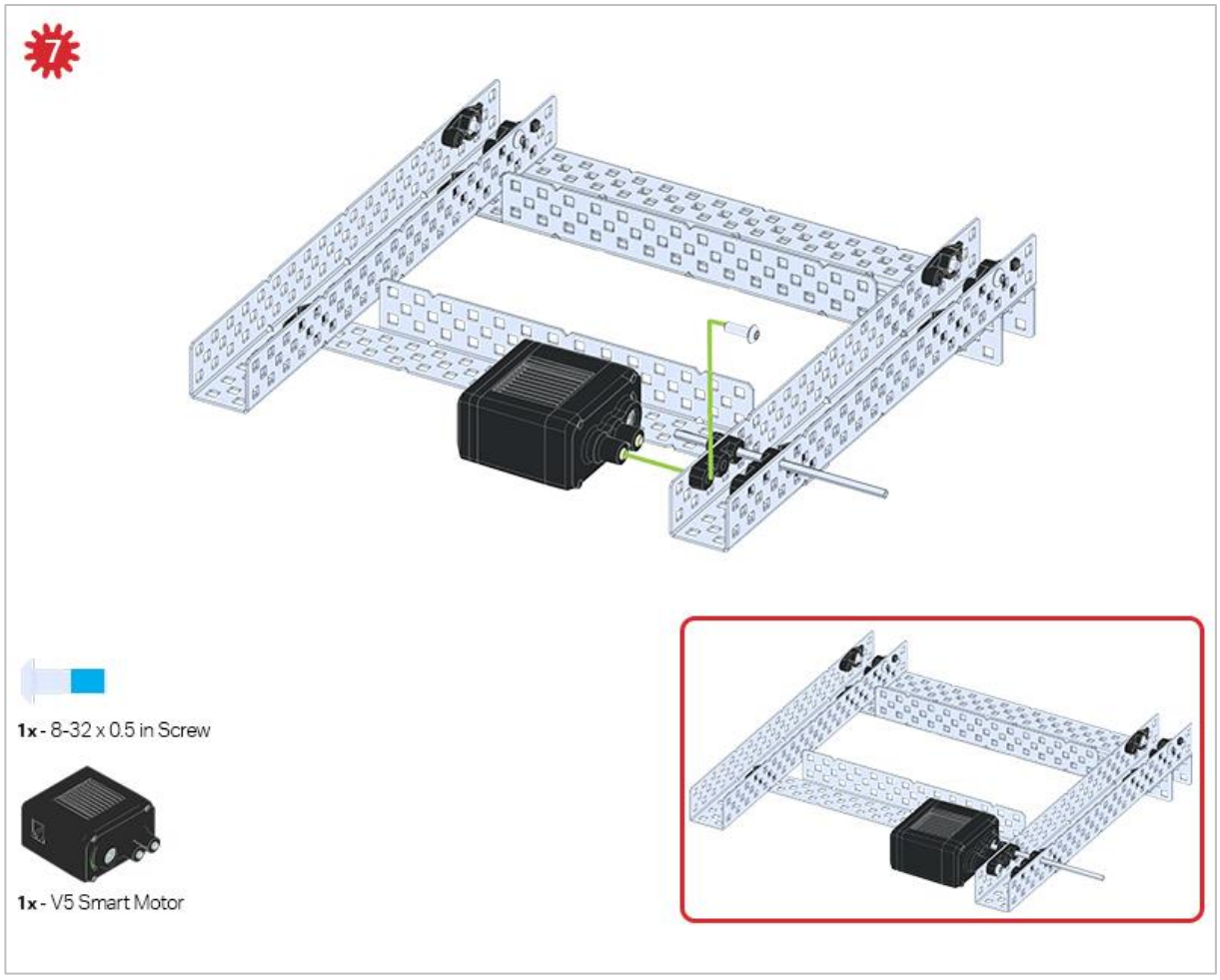
**2x** - 8-32 Nut

**2x** - 8-32 x 0.375 in Screw

**2x** - 1 Post Hex Nut Retainer w/ Bearing Flat

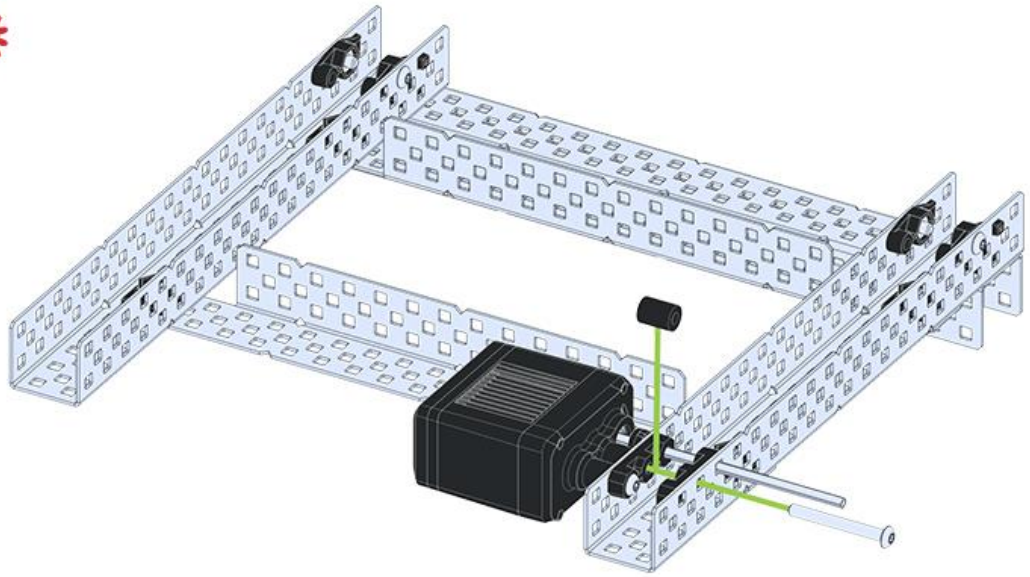



*Only one of the two sub-assemblies made in this step is used right now. The other will be used later in step 9.*



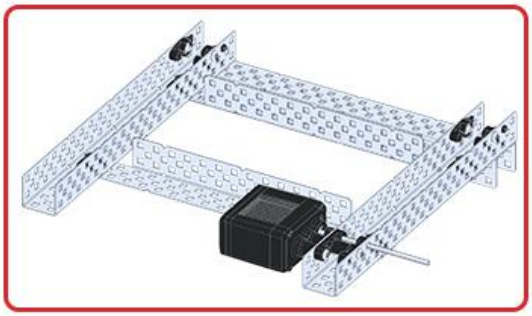

*Make sure your Smart Motors are oriented in the correct direction (screw holes facing the outside of the build and the shaft hole towards the inside).*

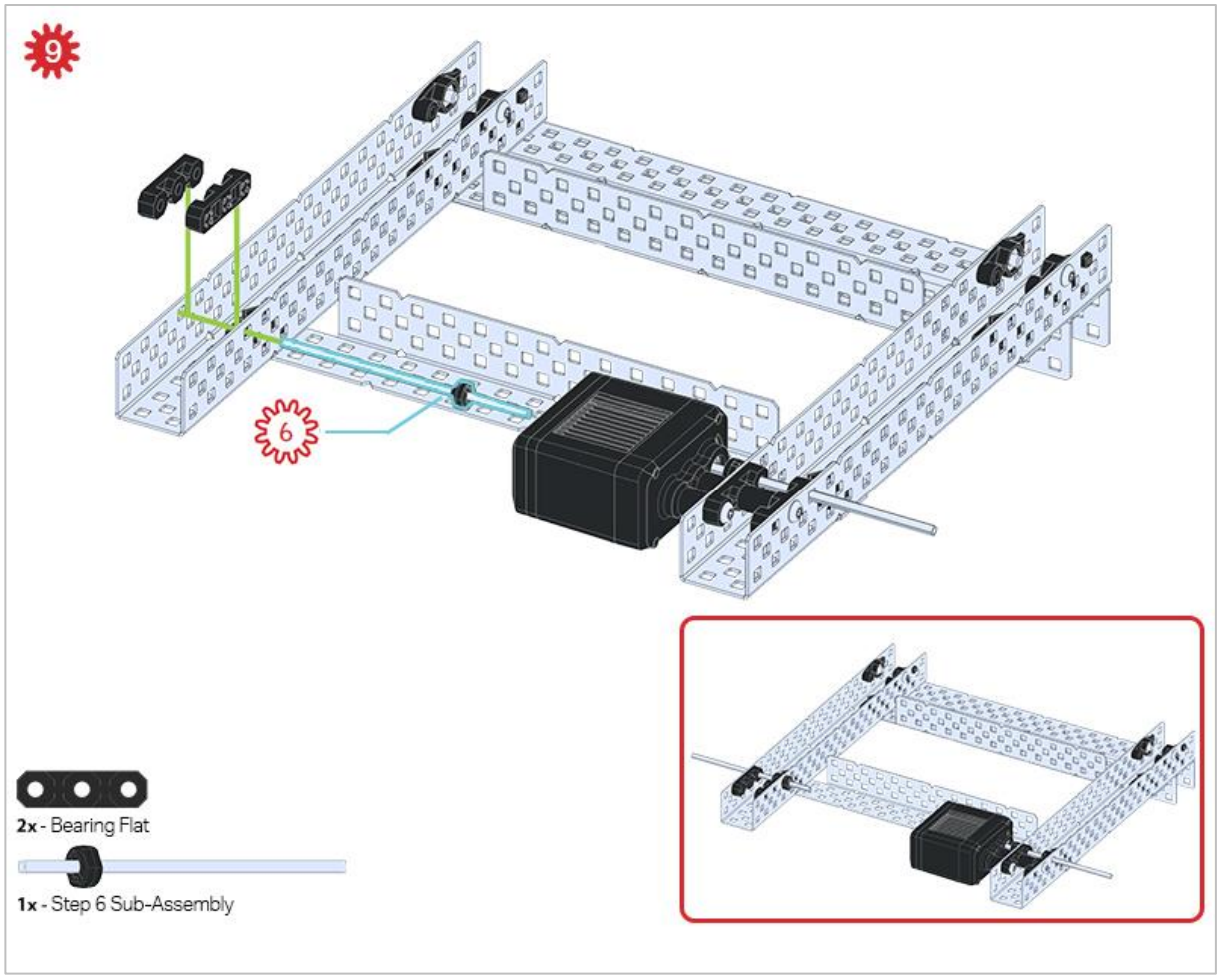
8

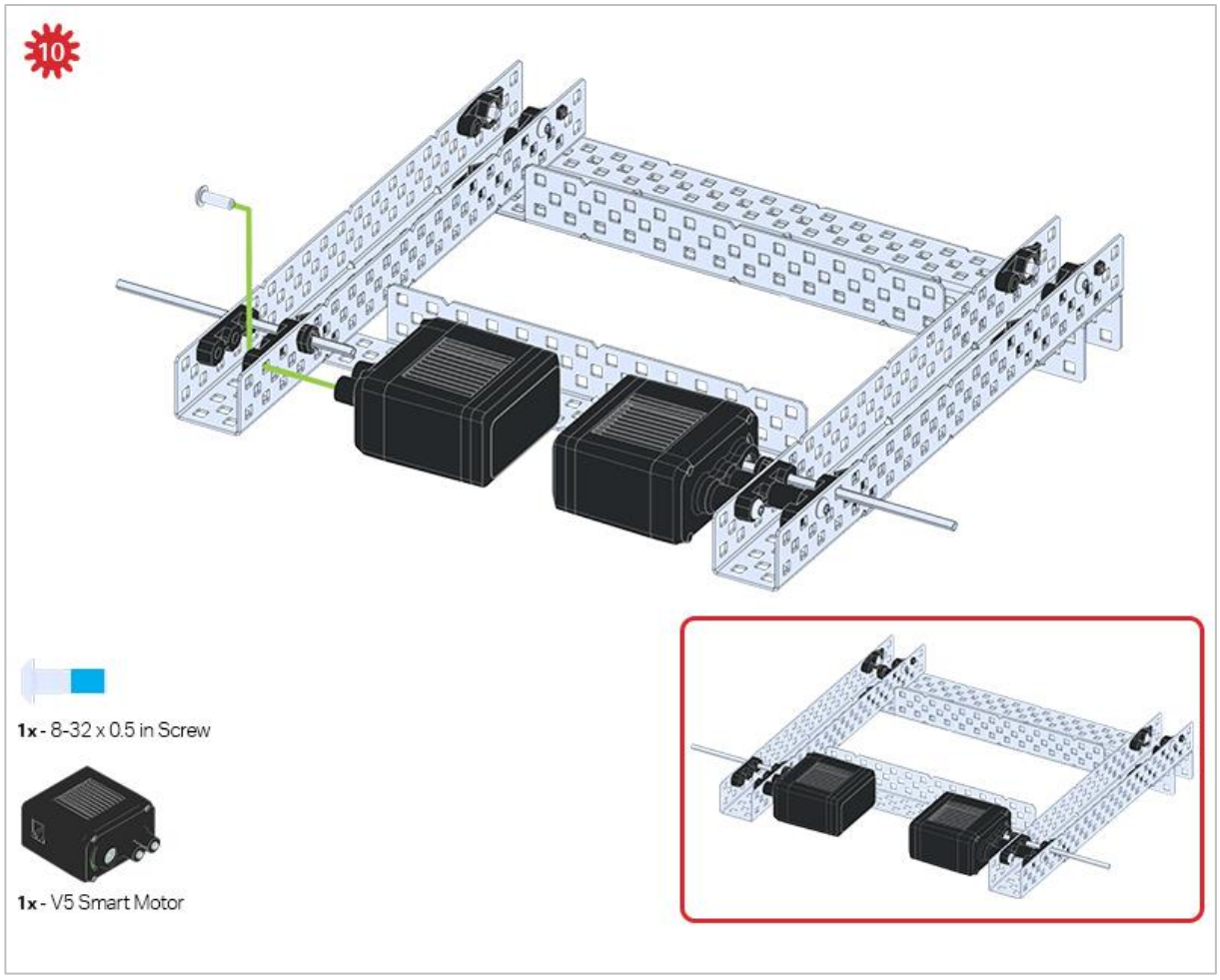


**1x** - 0.5 in Spacer

**1x** - 8-32 x 1.5 in Screw

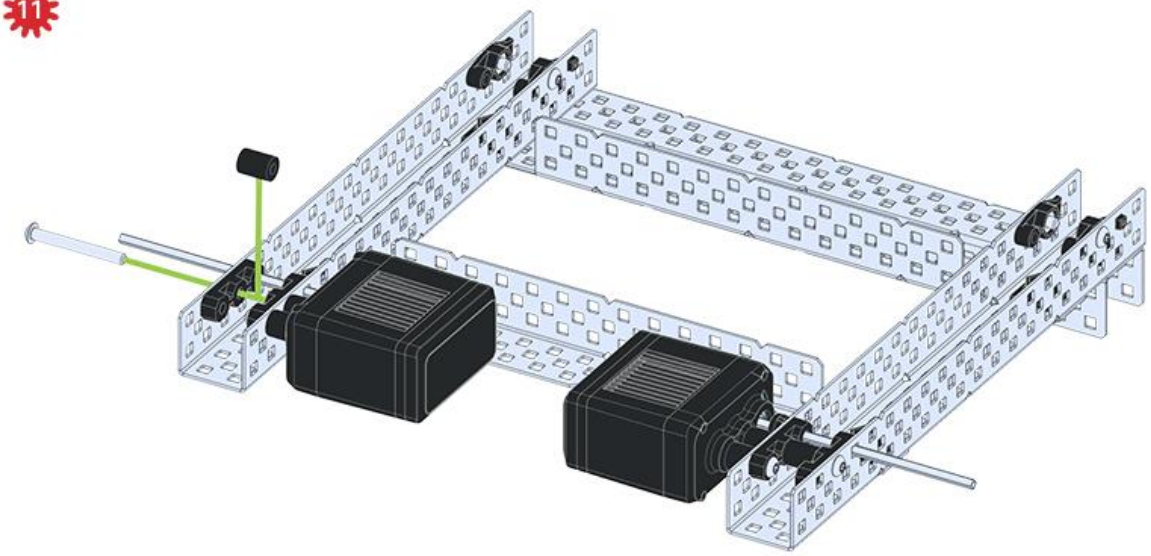



9
6
2x - Bearing Flat
1x - Step 6 Sub-Assembly



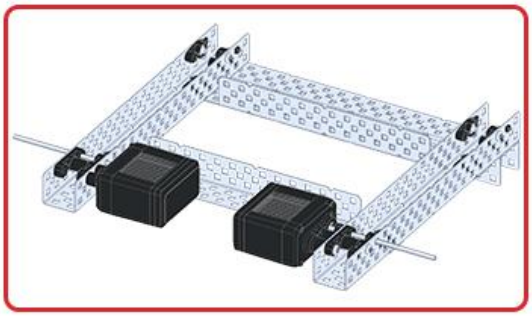

*Make sure your Smart Motors are oriented in the correct direction (screw holes facing the outside of the build and the shaft hole towards the inside).*

11

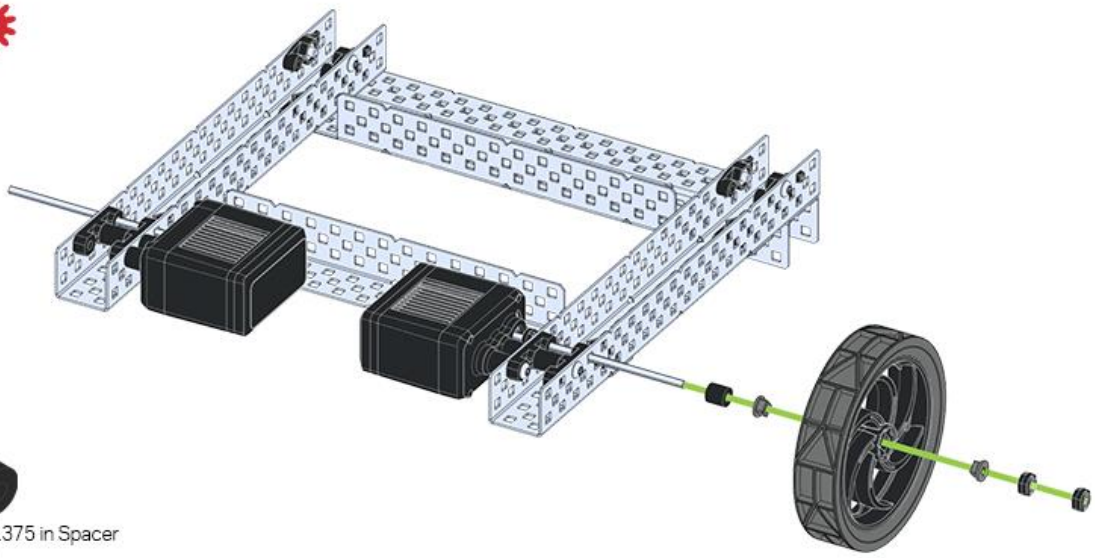


**1x** - 0.5 in Spacer

**1x** - 8-32 x 1.5 in Screw



12

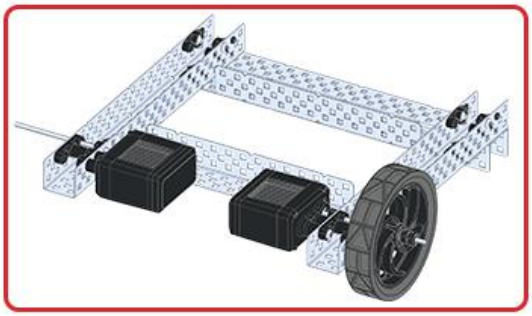


**1x** - 0.375 in Spacer

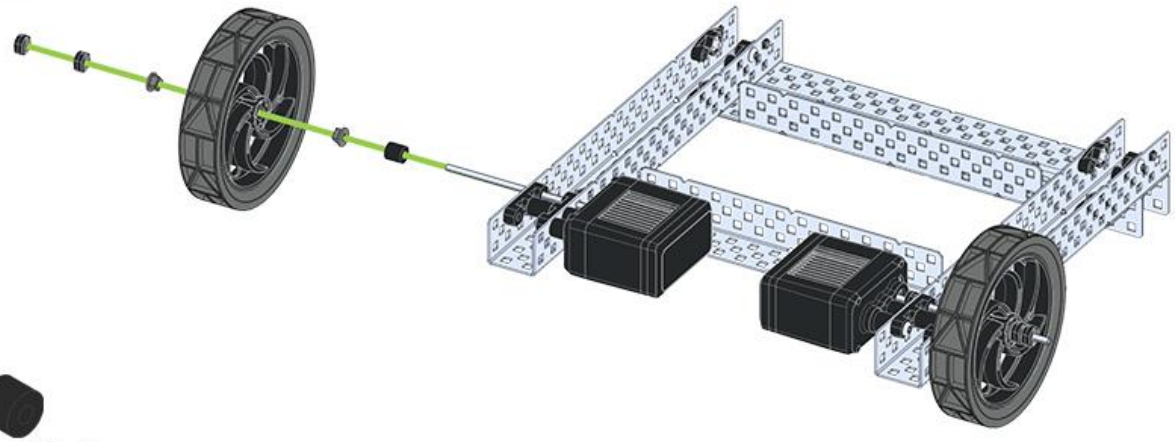
**2x** - Rubber Shaft Collar

**1x** - 4 in Wheel

**2x** - High Strength Shaft Insert



13



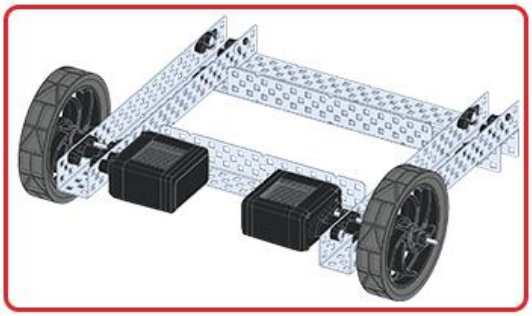


**1x** - 0.375 in Spacer

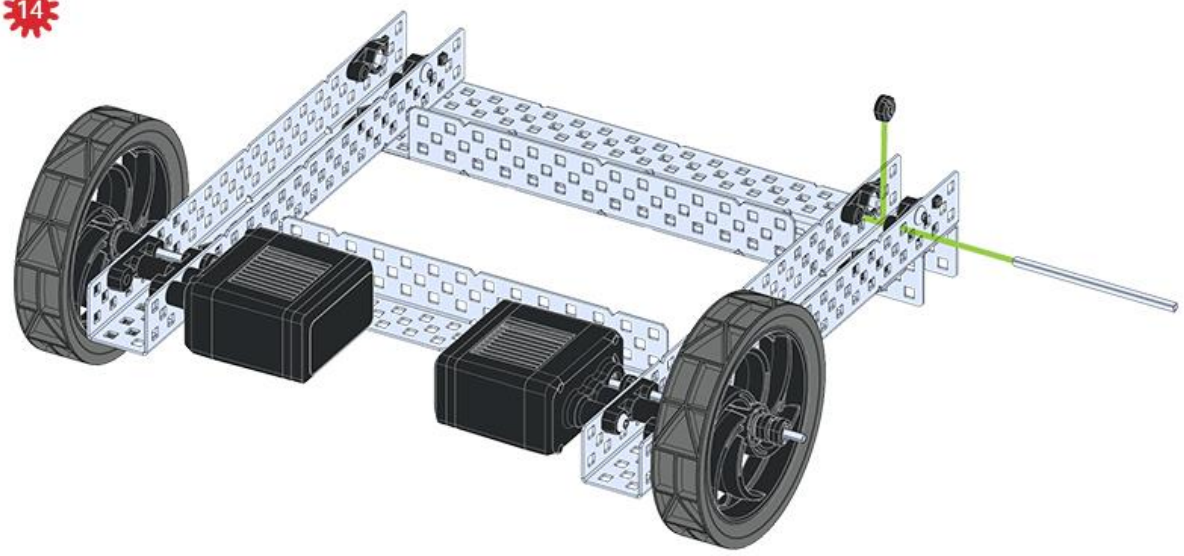
**2x** - Rubber Shaft Collar

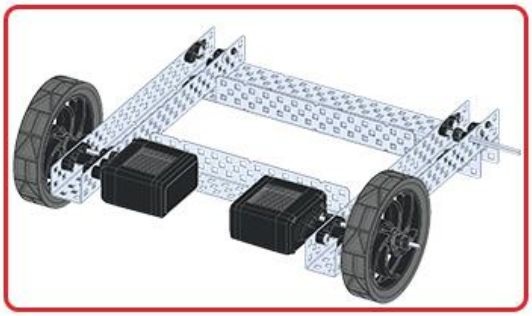
**1x** - 4 in Wheel

**2x** - High Strength Shaft Insert



14

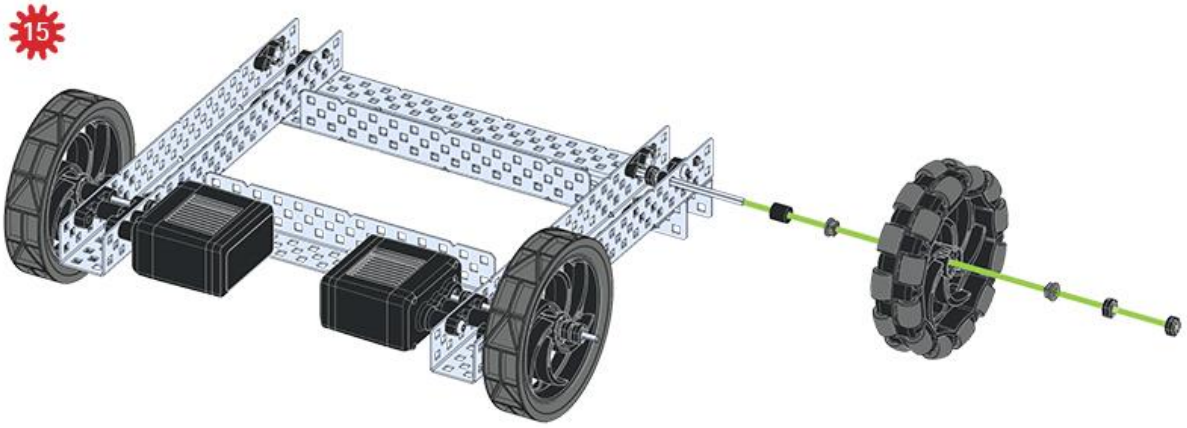




**1x** - Rubber Shaft Collar

**1x** - 3 in Shaft

15

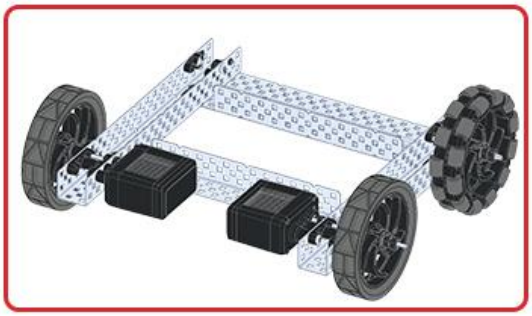


**1x** - 0.375 in Spacer

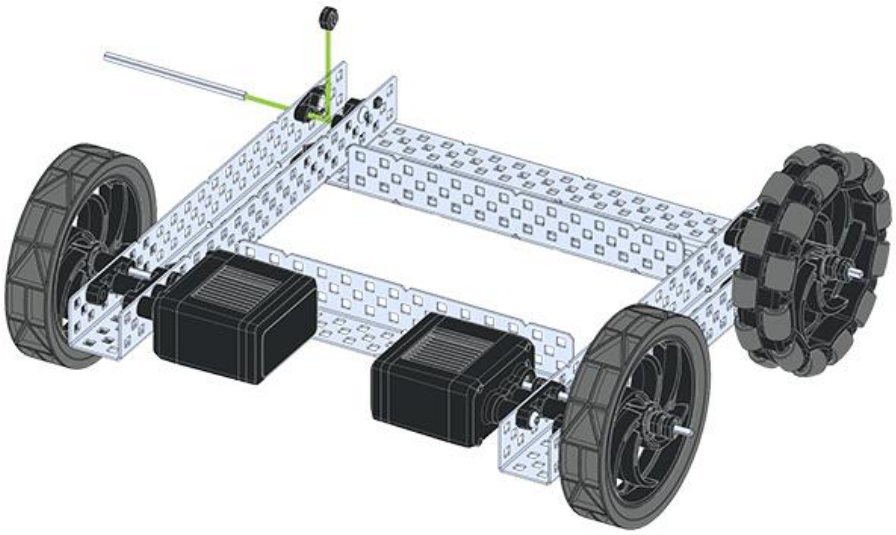
**2x** - Rubber Shaft Collar

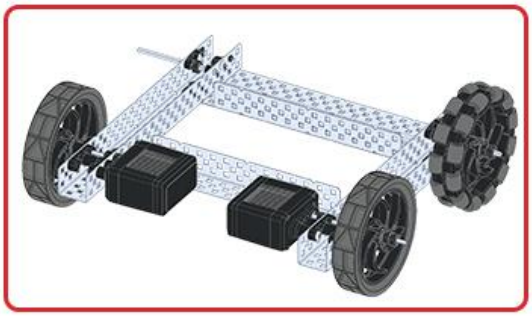
**1x** - 4 in Omni Wheel

**2x** - High Strength Shaft Insert



16

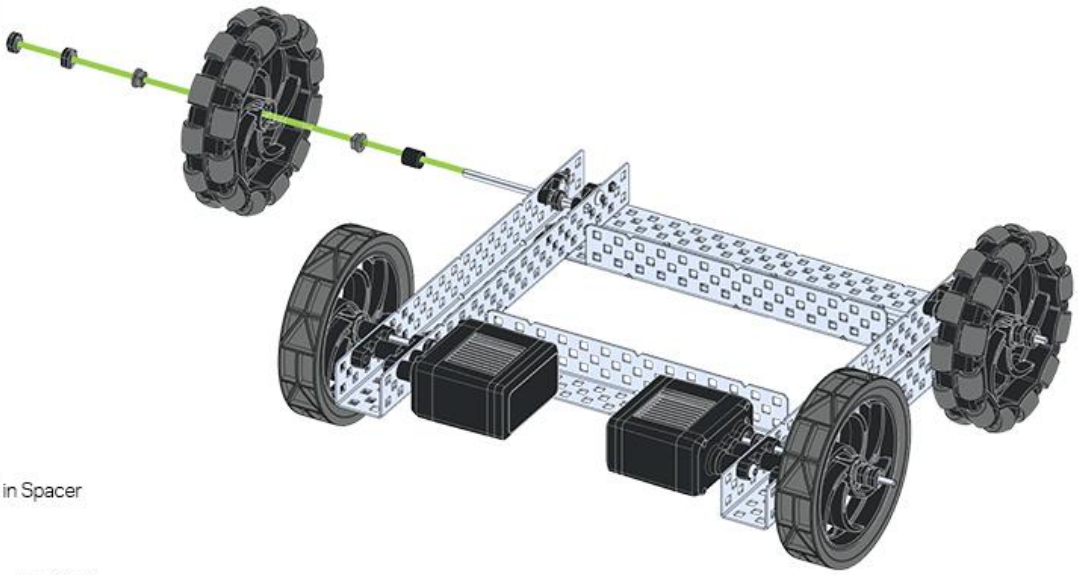




**1x** - Rubber Shaft Collar

**1x** - 3 in Shaft

17

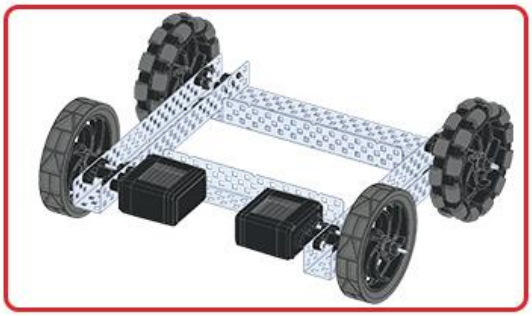


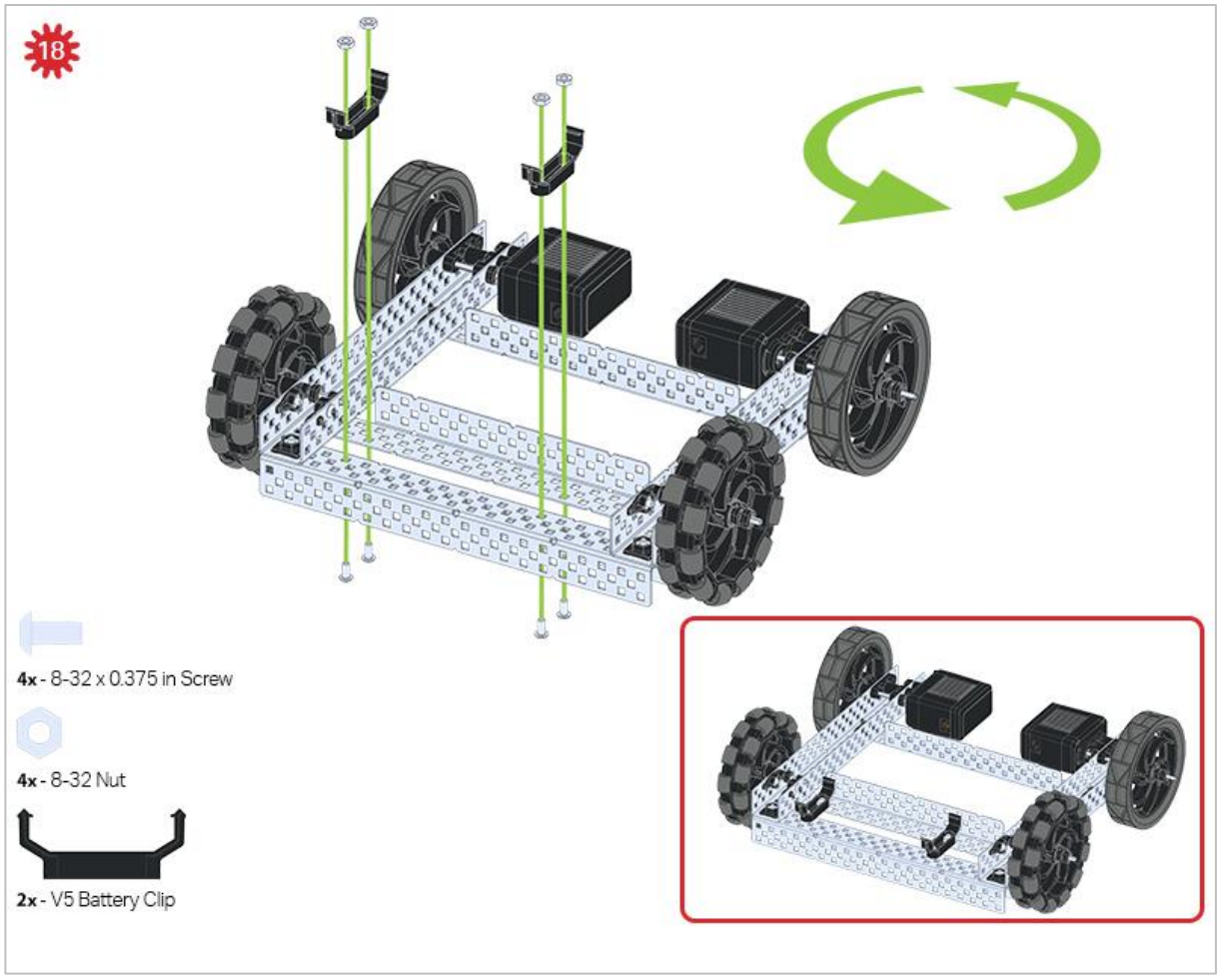
**1x** - 0.375 in Spacer

**2x** - Rubber Shaft Collar

**1x** - 4 in Omni Wheel

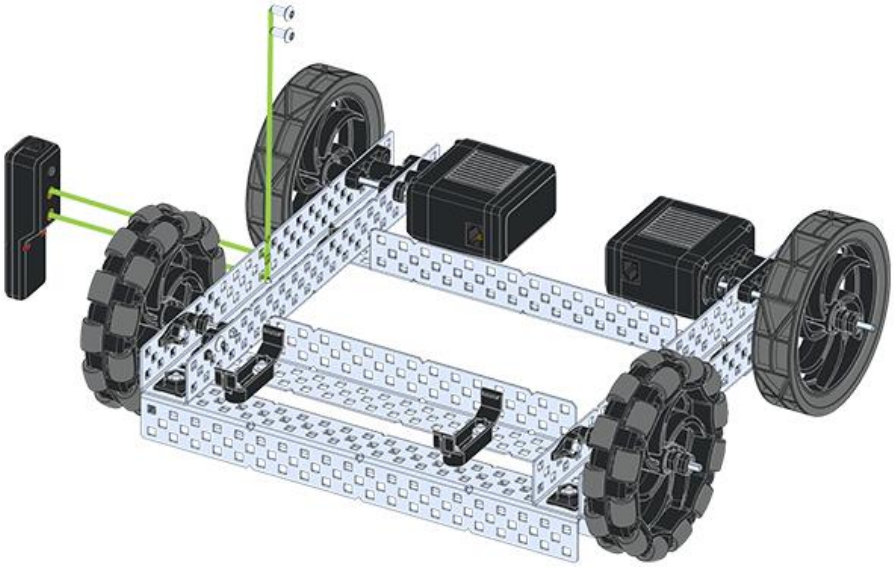
**2x** - High Strength Shaft Insert






*The green icon indicates that the build needs to be rotated (180 degrees).*

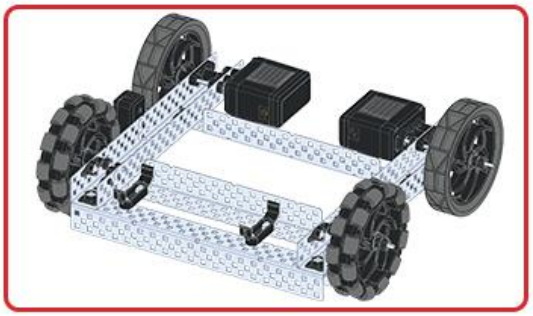
19

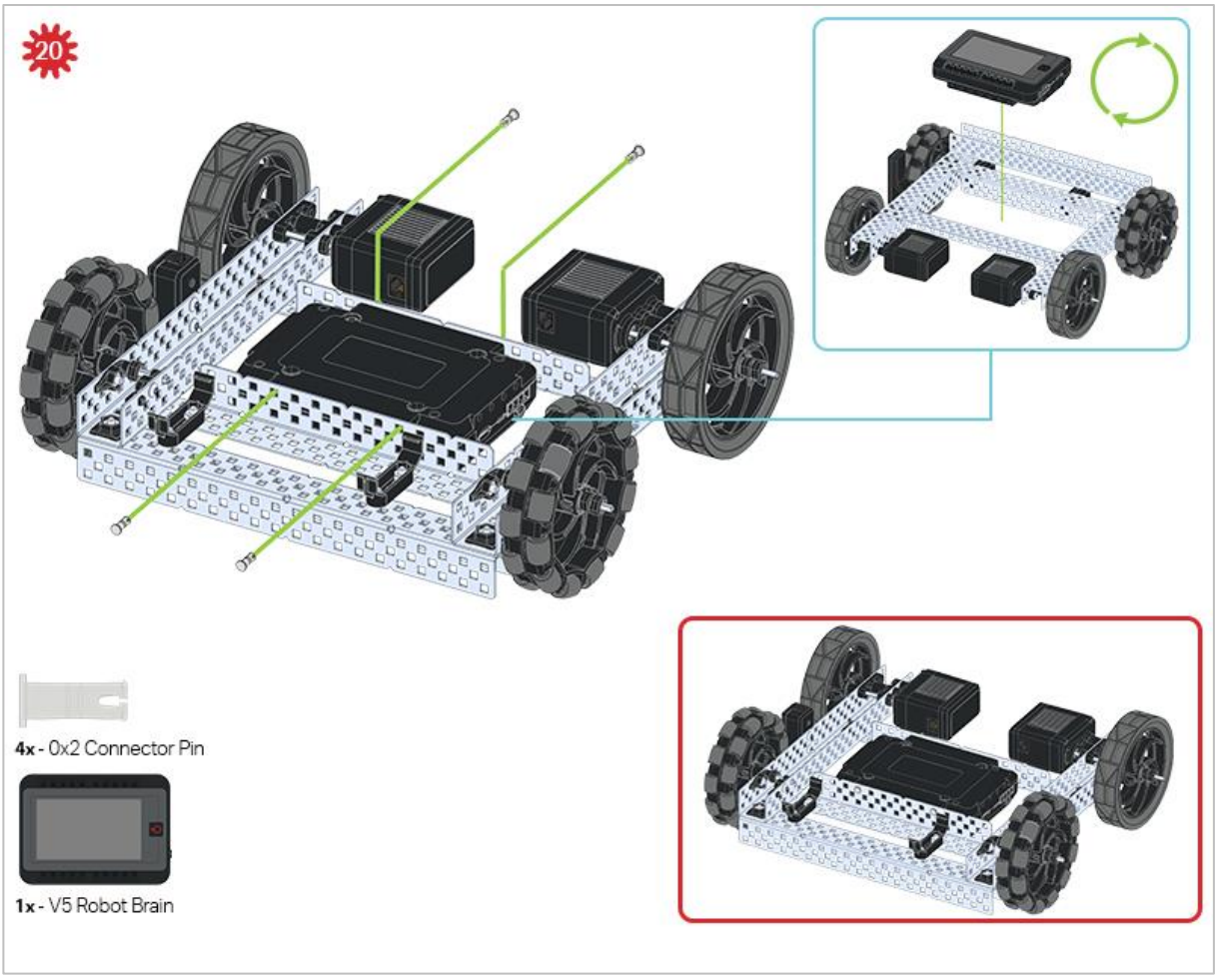


**1x** - V5 Radio

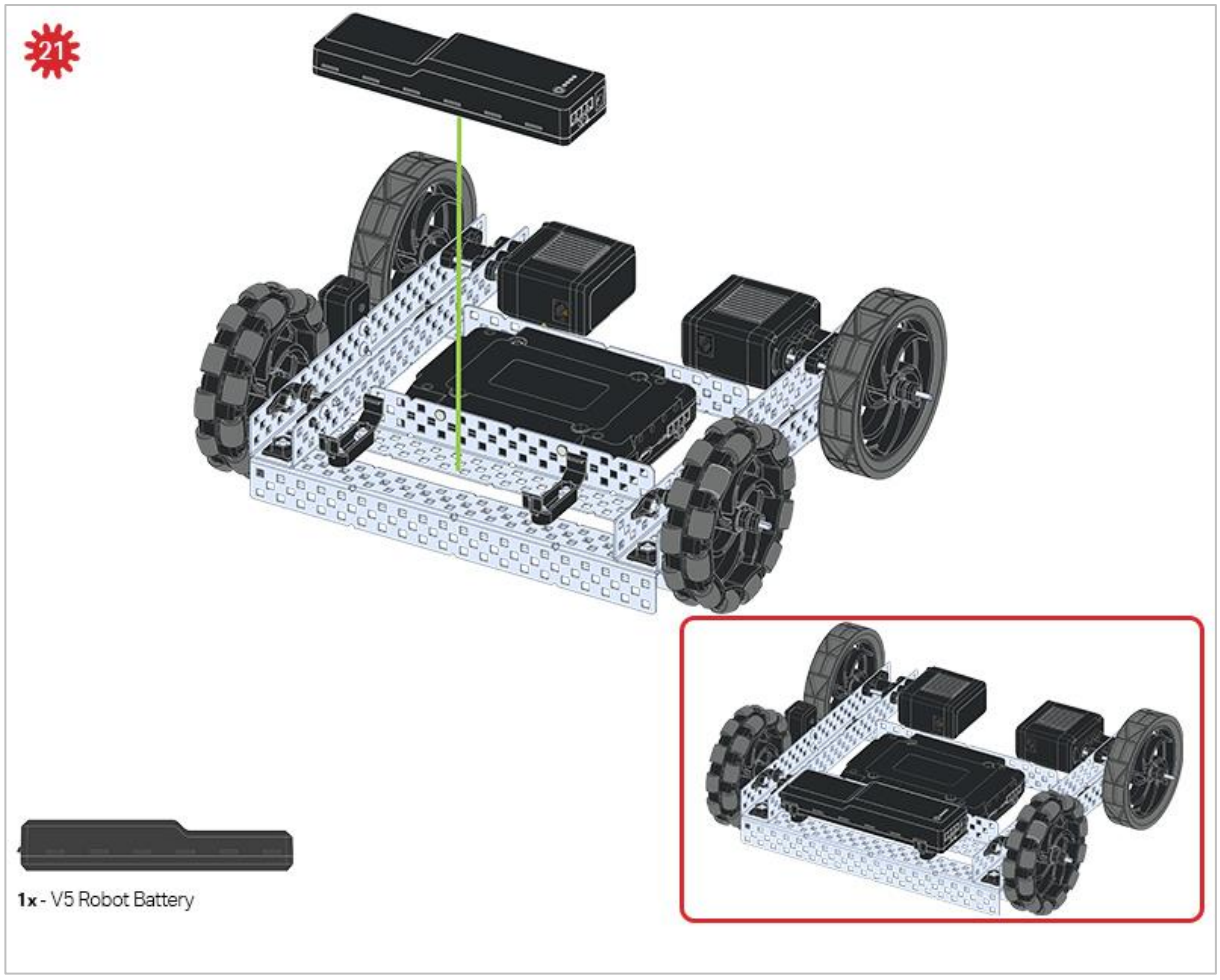


**2x** - 8-32 x 0.375 in Screw

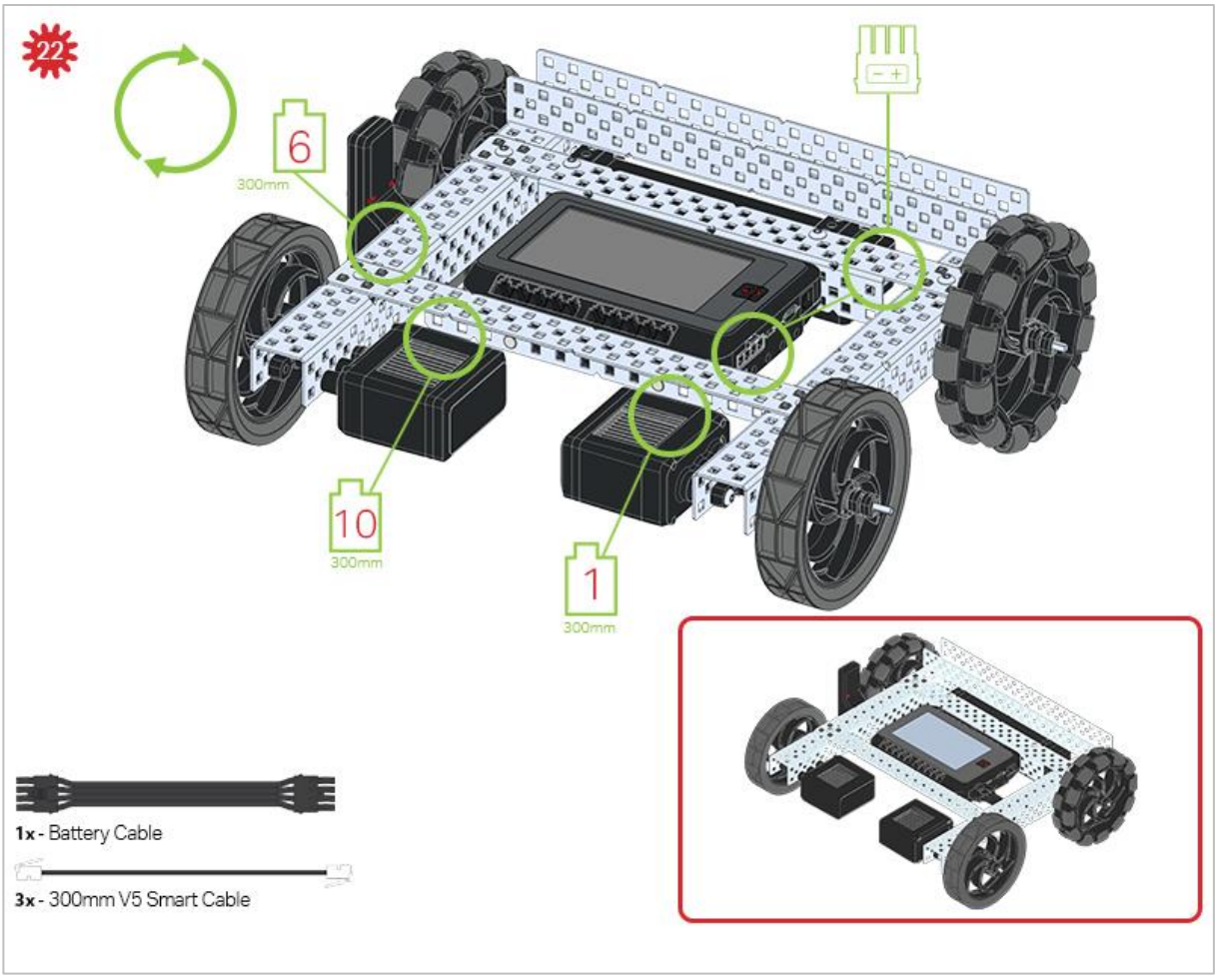




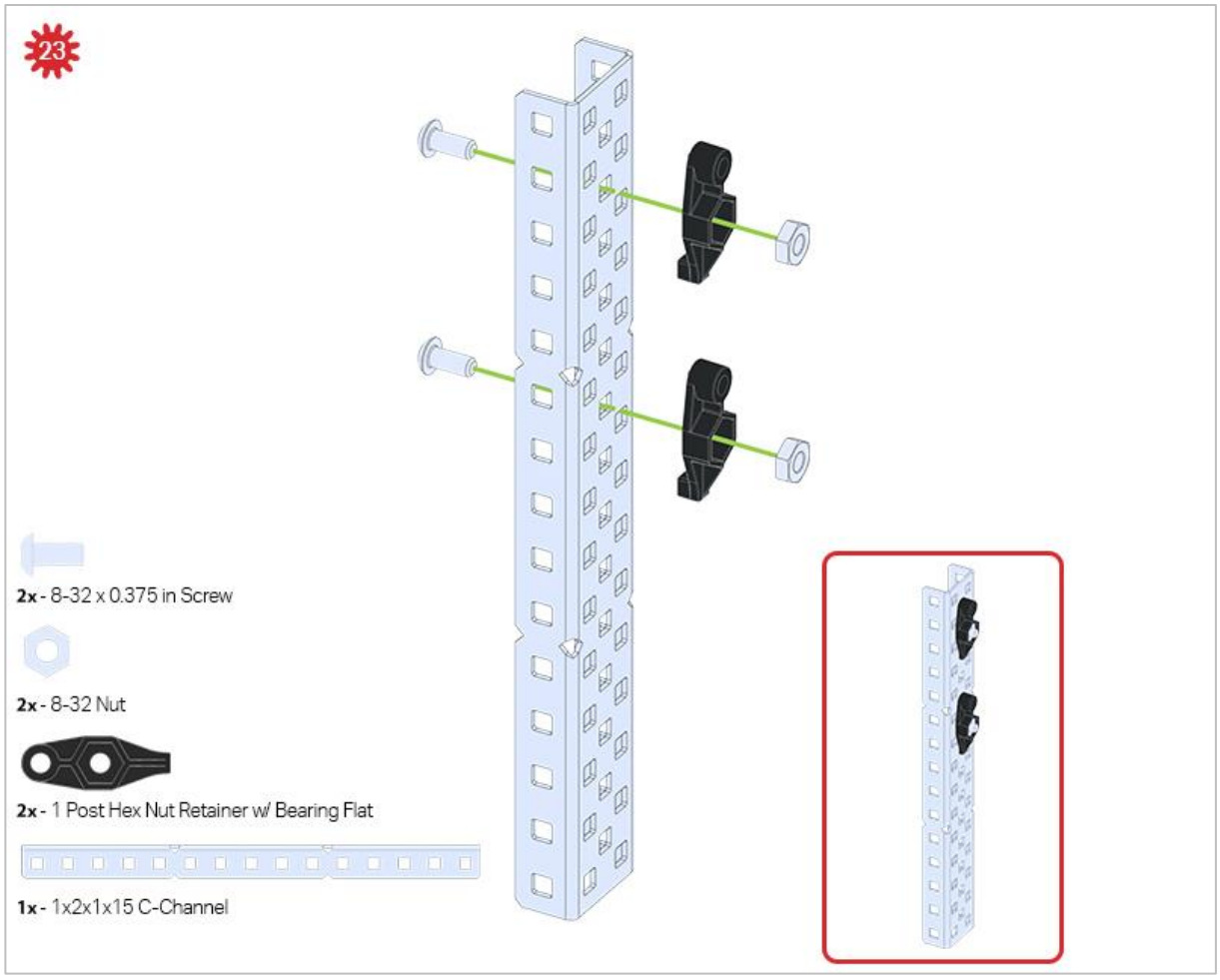

*The blue call out shows what the orientation of the Robot Brain should be if the build were flipped right side up. Make sure the 3 wire ports on the Robot Brain are facing the V5 Radio!*

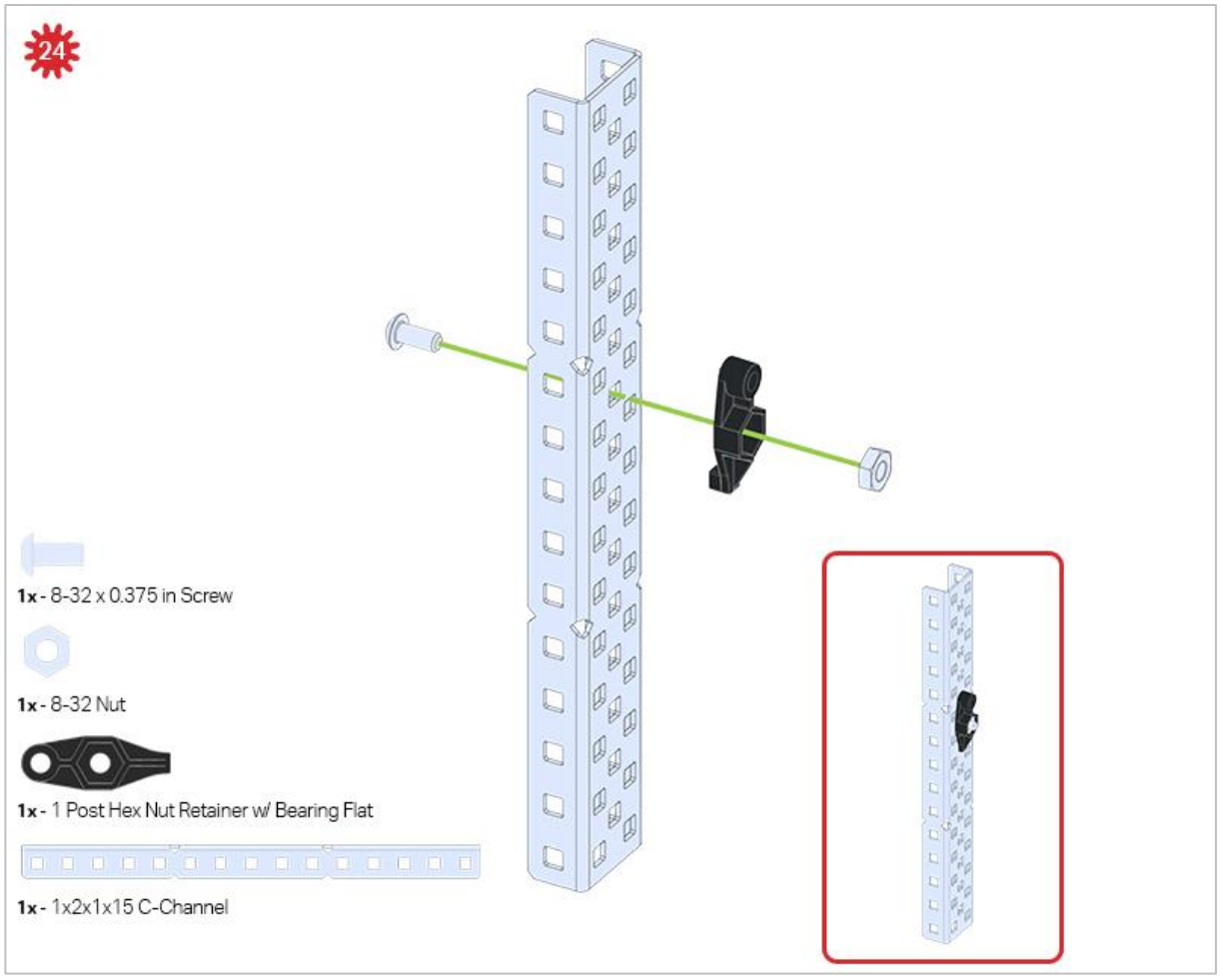



**1x** - V5 Robot Battery

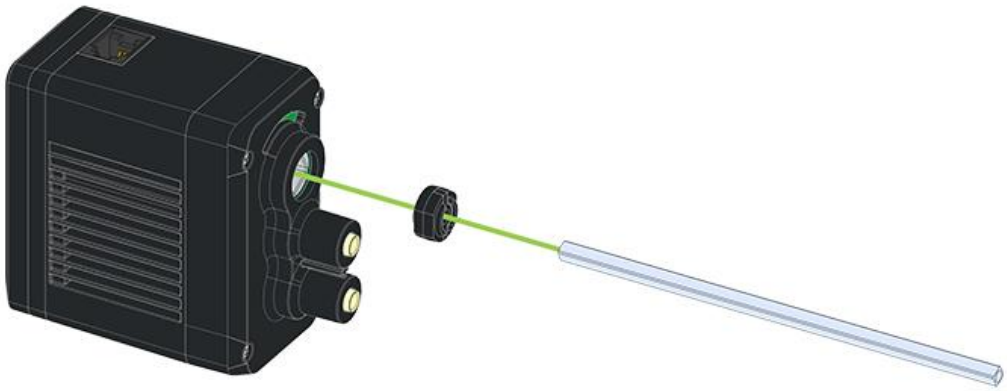



*The green call outs indicate which port on the Robot Brain to plug each device into using their respective cable.*


23
2x - 8-32 x 0.375 in Screw
2x - 8-32 Nut
2x - 1 Post Hex Nut Retainer w/ Bearing Flat
1x - 1x2x1x15 C-Channel


24
1x - 8-32 x 0.375 in Screw
1x - 8-32 Nut
1x - 1 Post Hex Nut Retainer w/ Bearing Flat
1x - 1x2x1x15 C-Channel

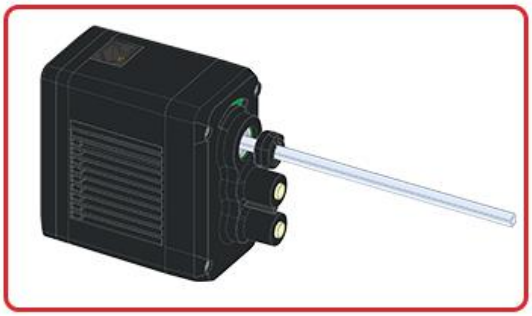
25

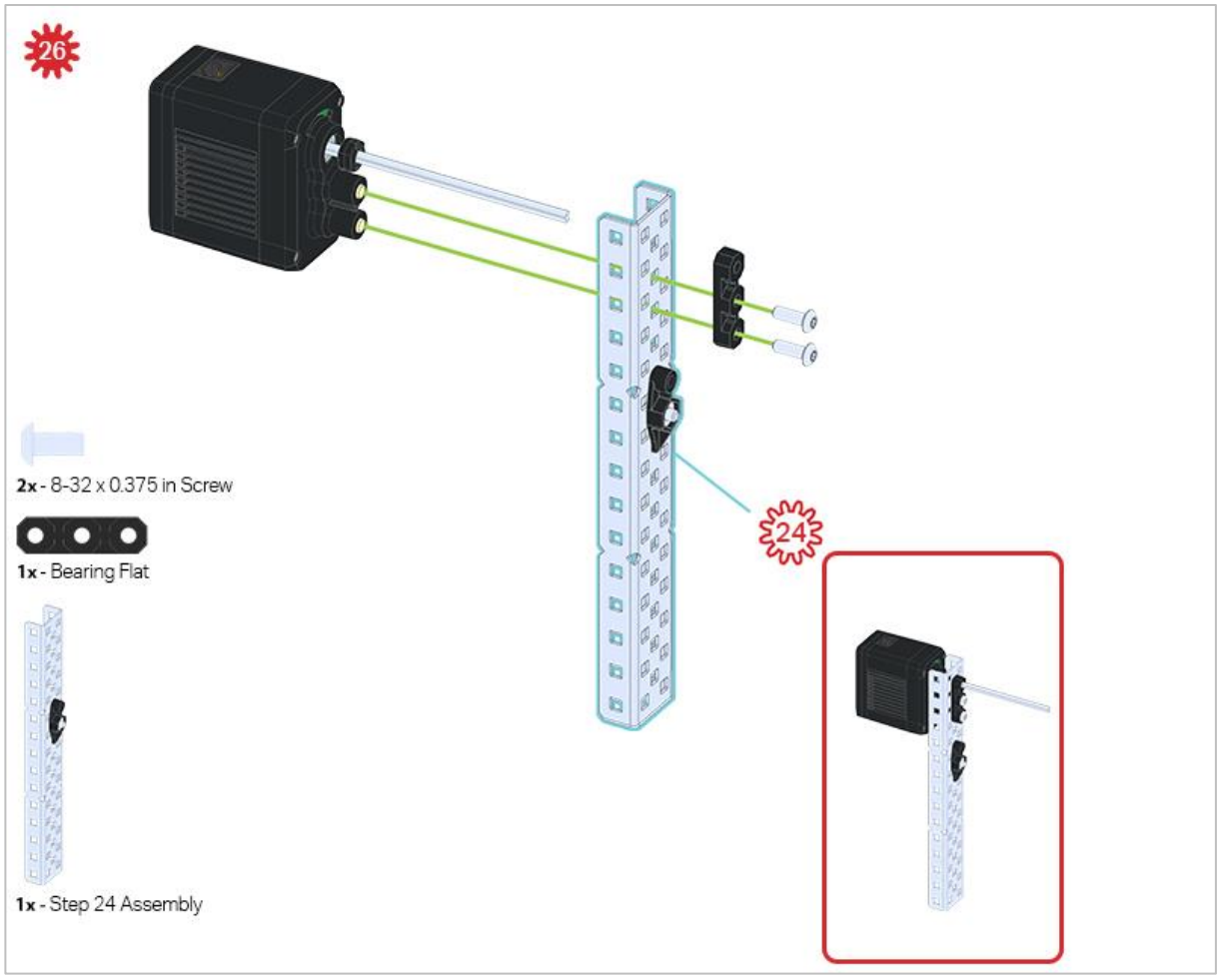


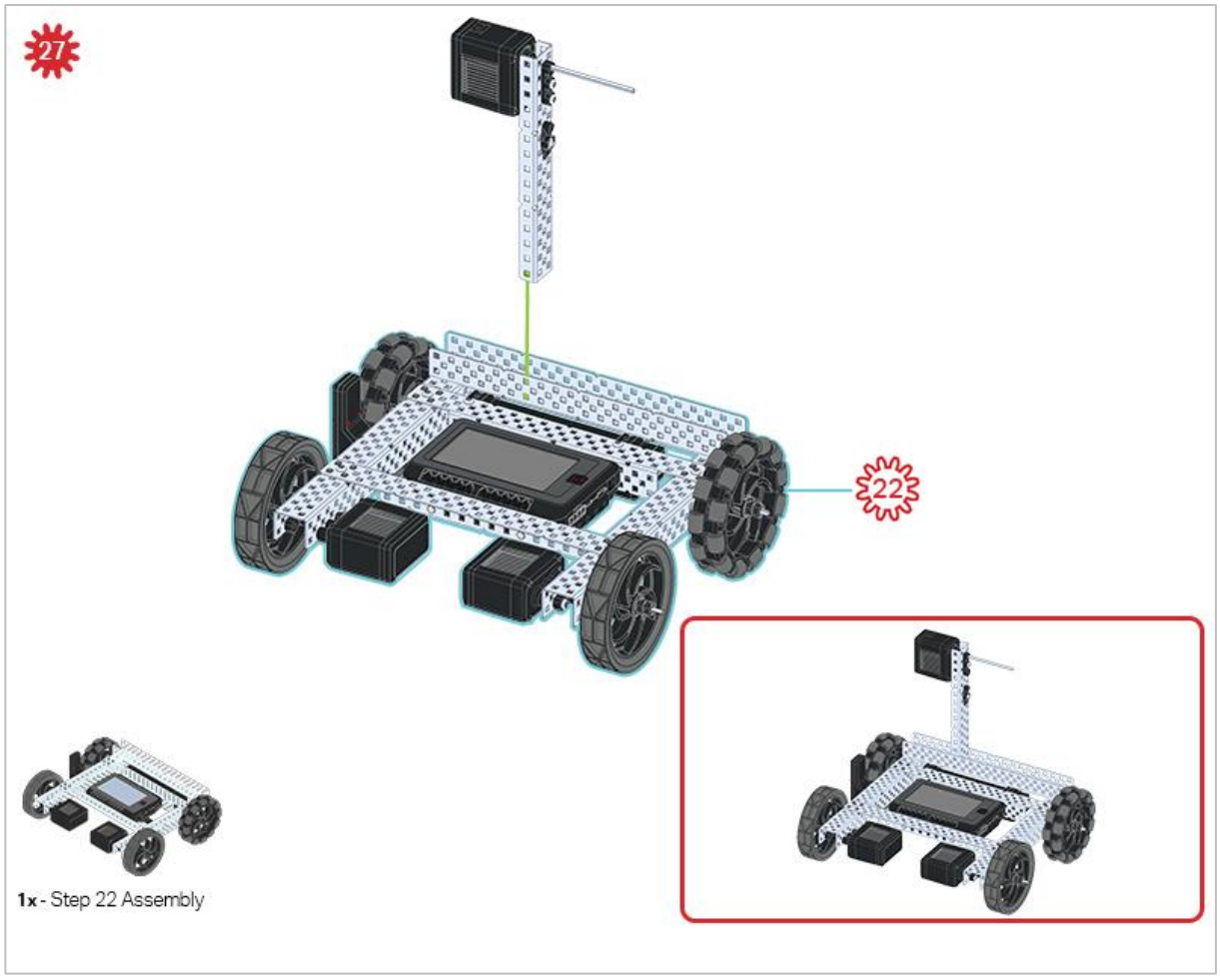
**1x** - Rubber Shaft Collar

**1x** - V5 Smart Motor

**1x** - 4 in Shaft




26
2x - 8-32 x 0.375 in Screw
1x - Bearing Flat
1x - Step 24 Assembly
24


27
22
1x - Step 22 Assembly

28

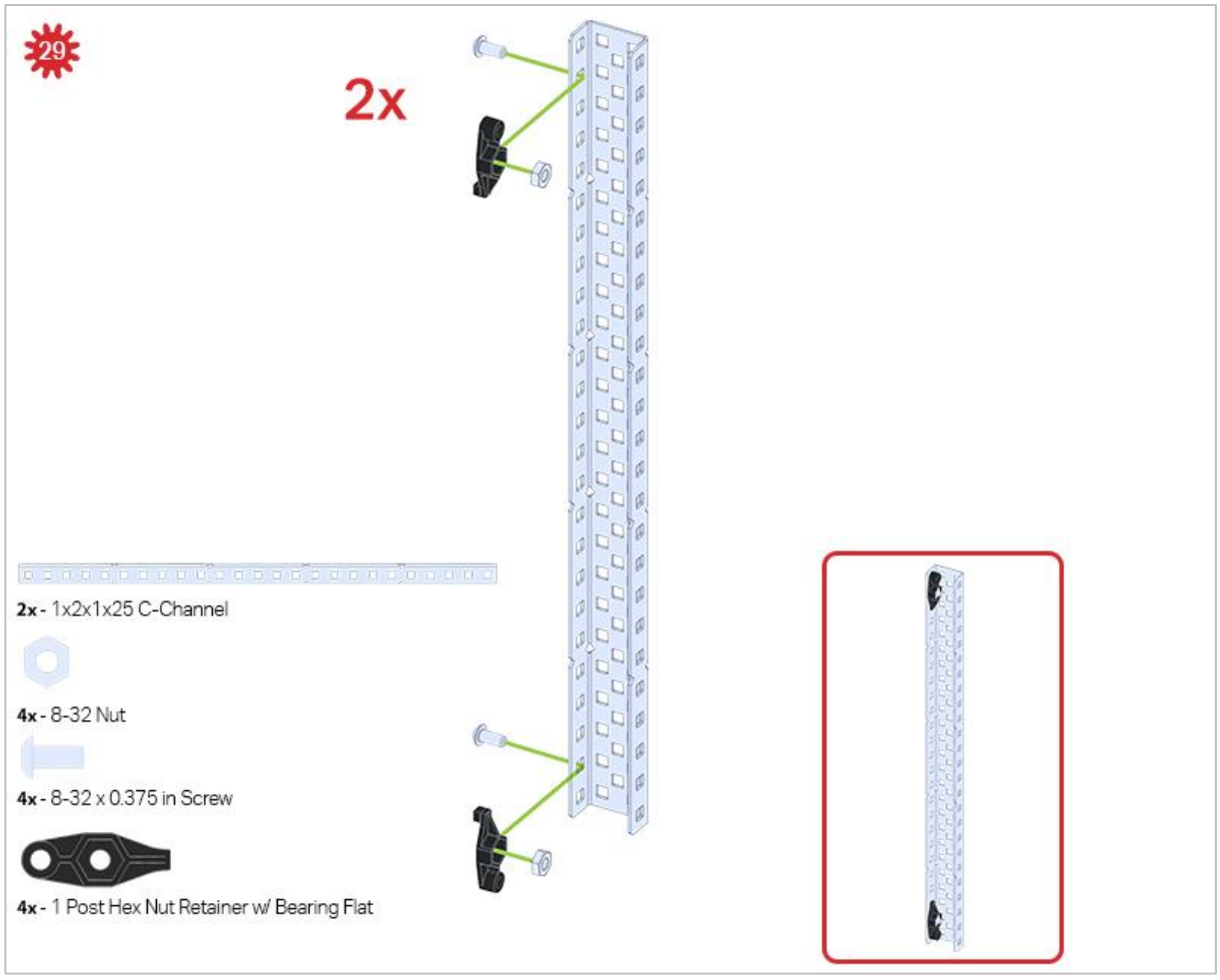


**1x** - 8-32 Nut

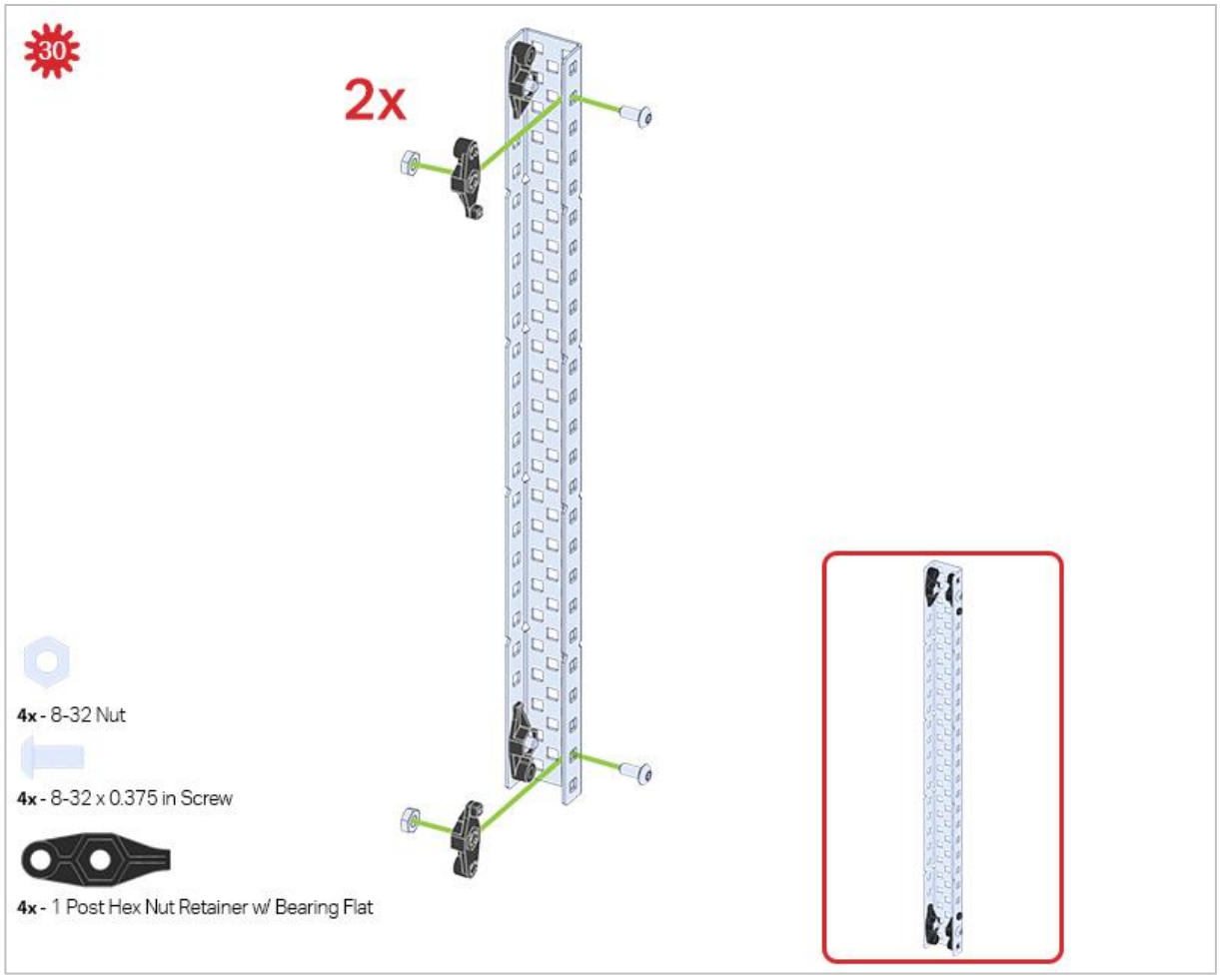
**1x** - 0.875 in Spacer

**2x** - 1 Post Hex Nut Retainer

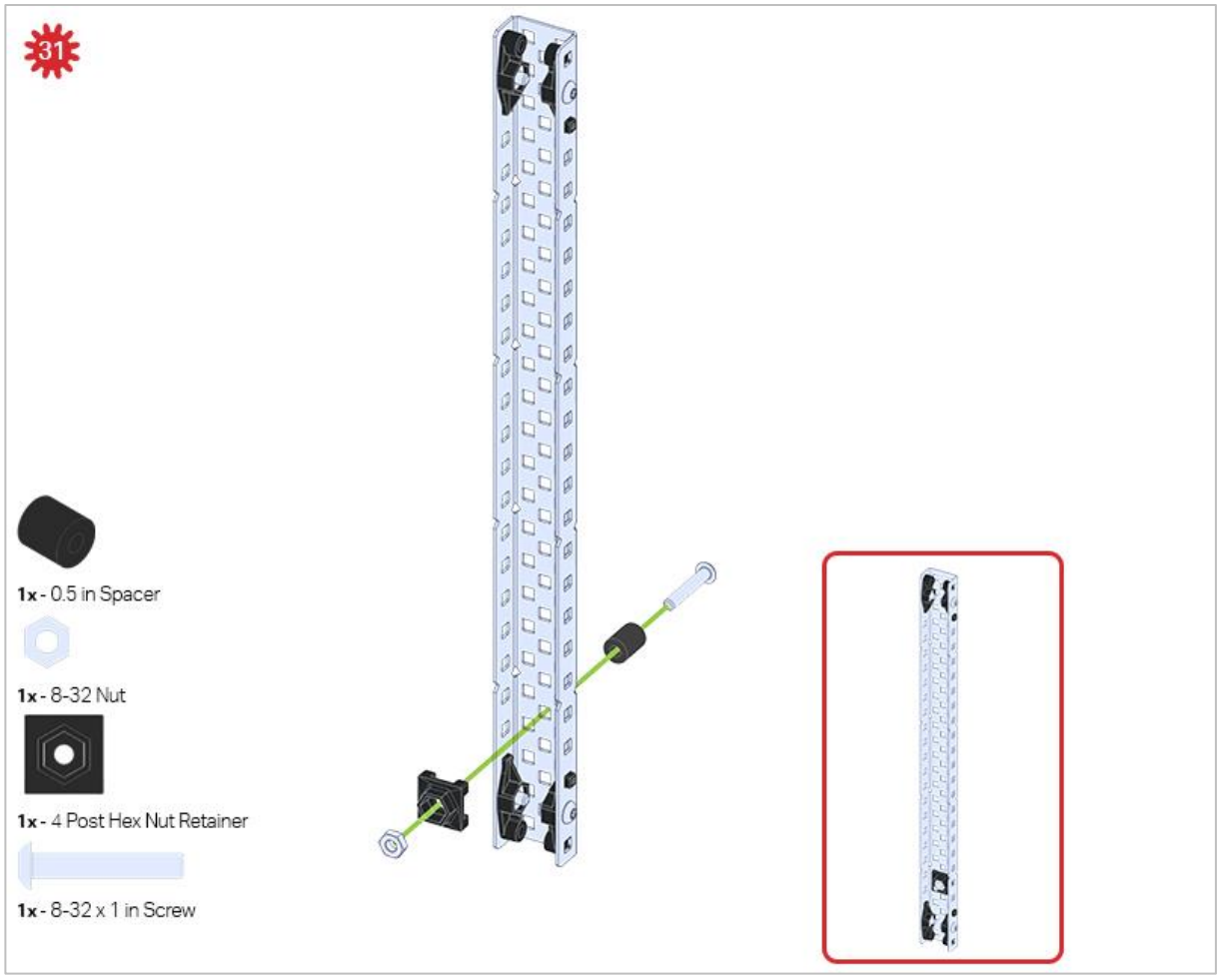
**1x** - 8-32 x 1.5 in Screw




*Be sure to make two assemblies in this step!*




*This step adds onto the two assemblies started in Step 29.*



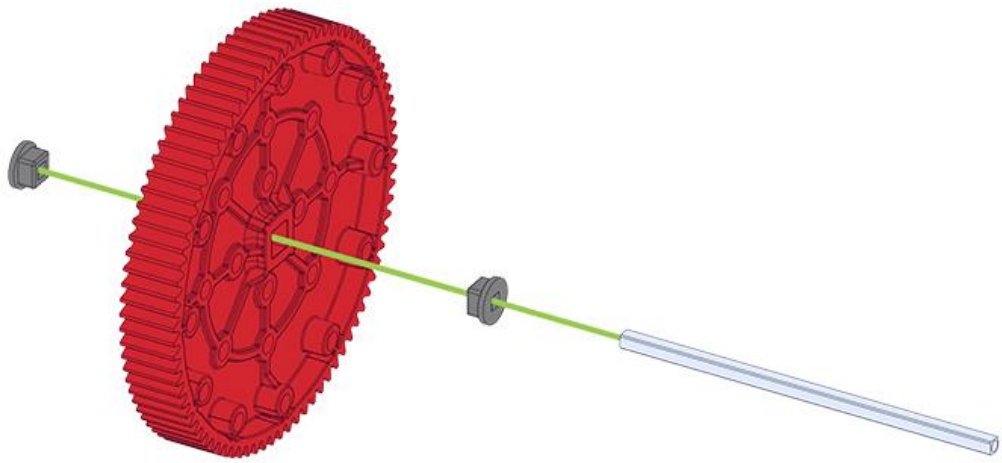

*Make sure to add this to only one of the two sub-assemblies you just made.*

32

**1x** - 8-32 Nut

**1x** - 1 Post Hex Nut Retainer

33

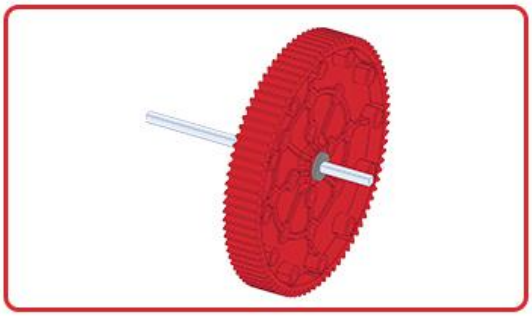


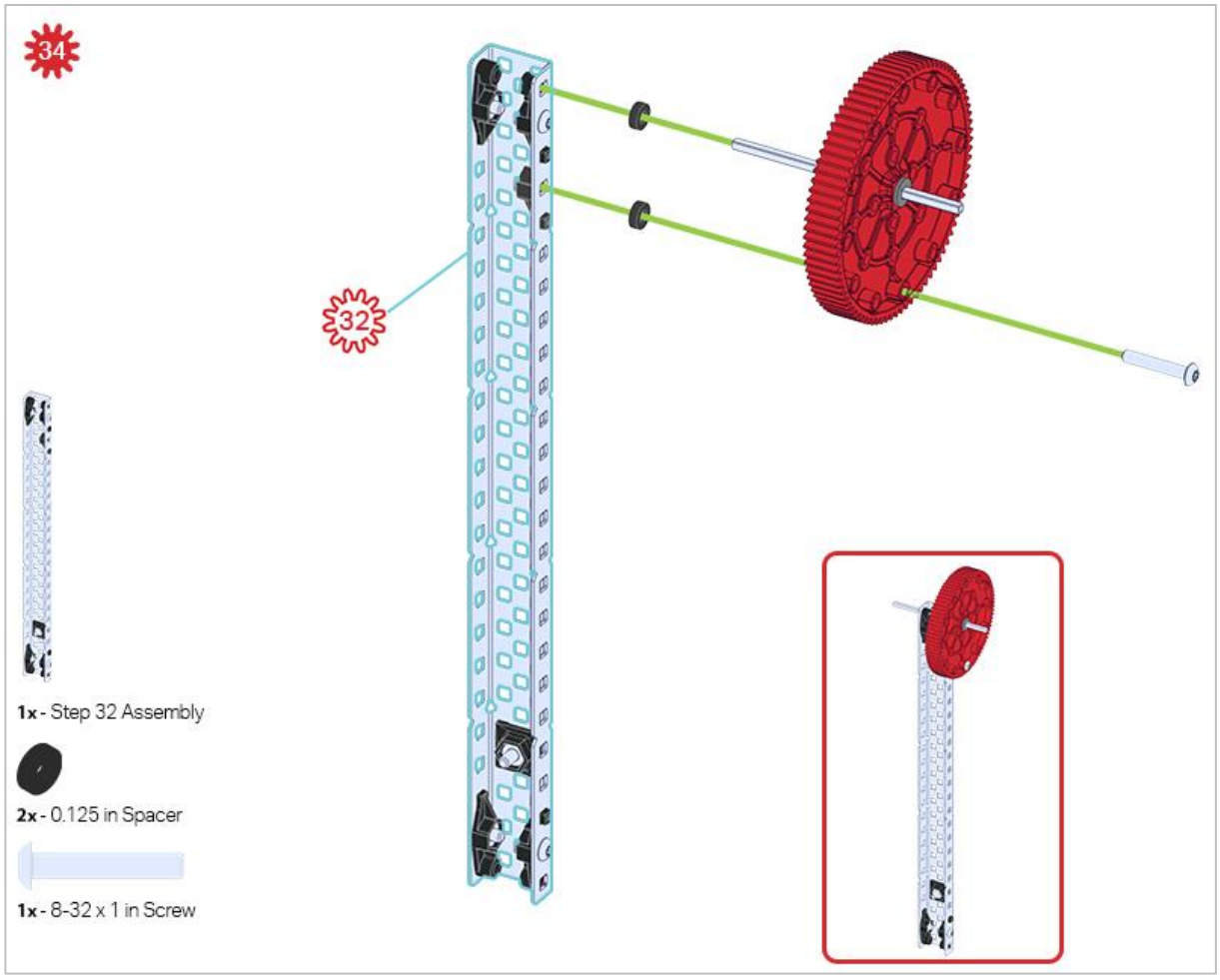
**1x** - High Strength 84 Tooth Gear

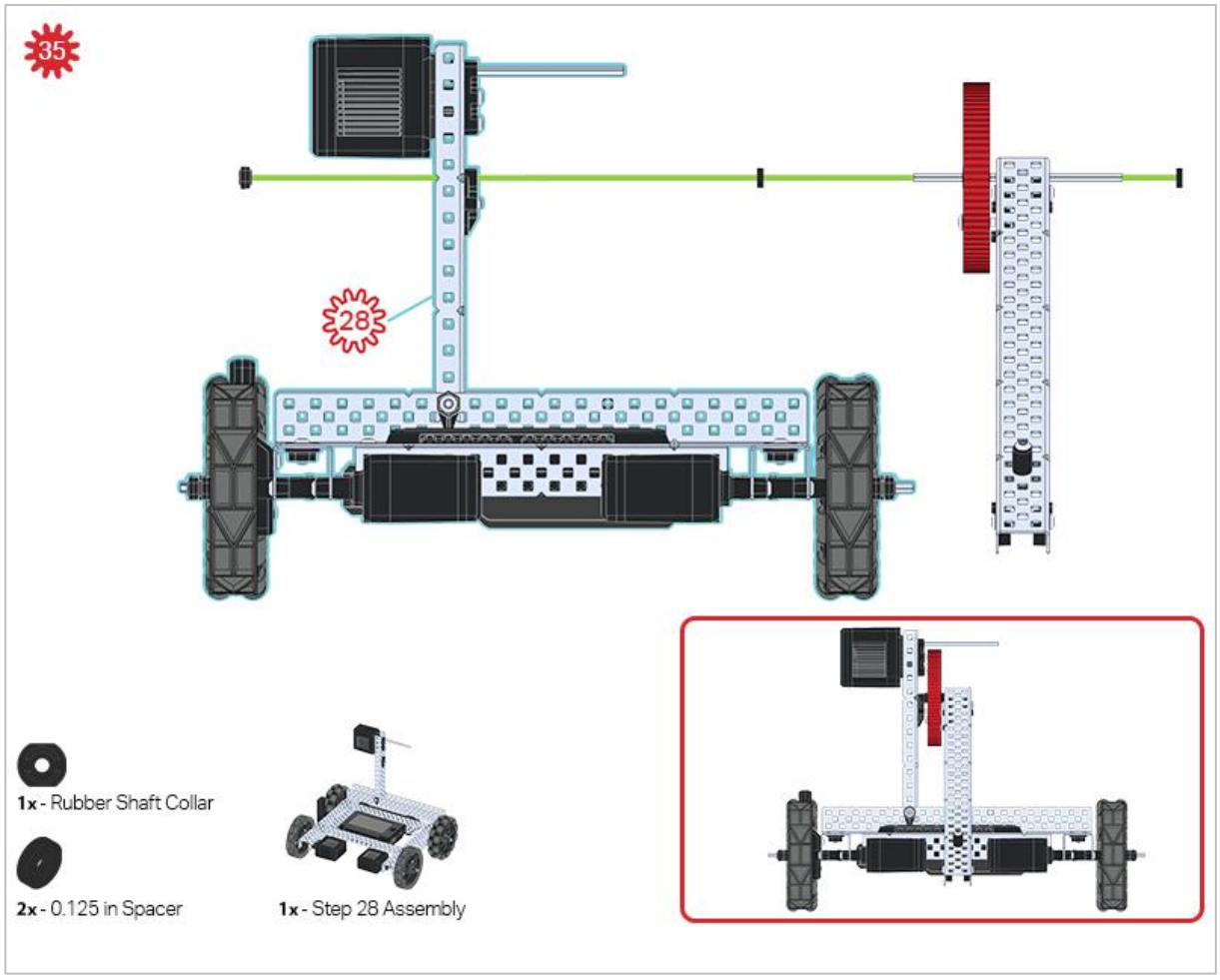
**2x** - High Strength Shaft Insert

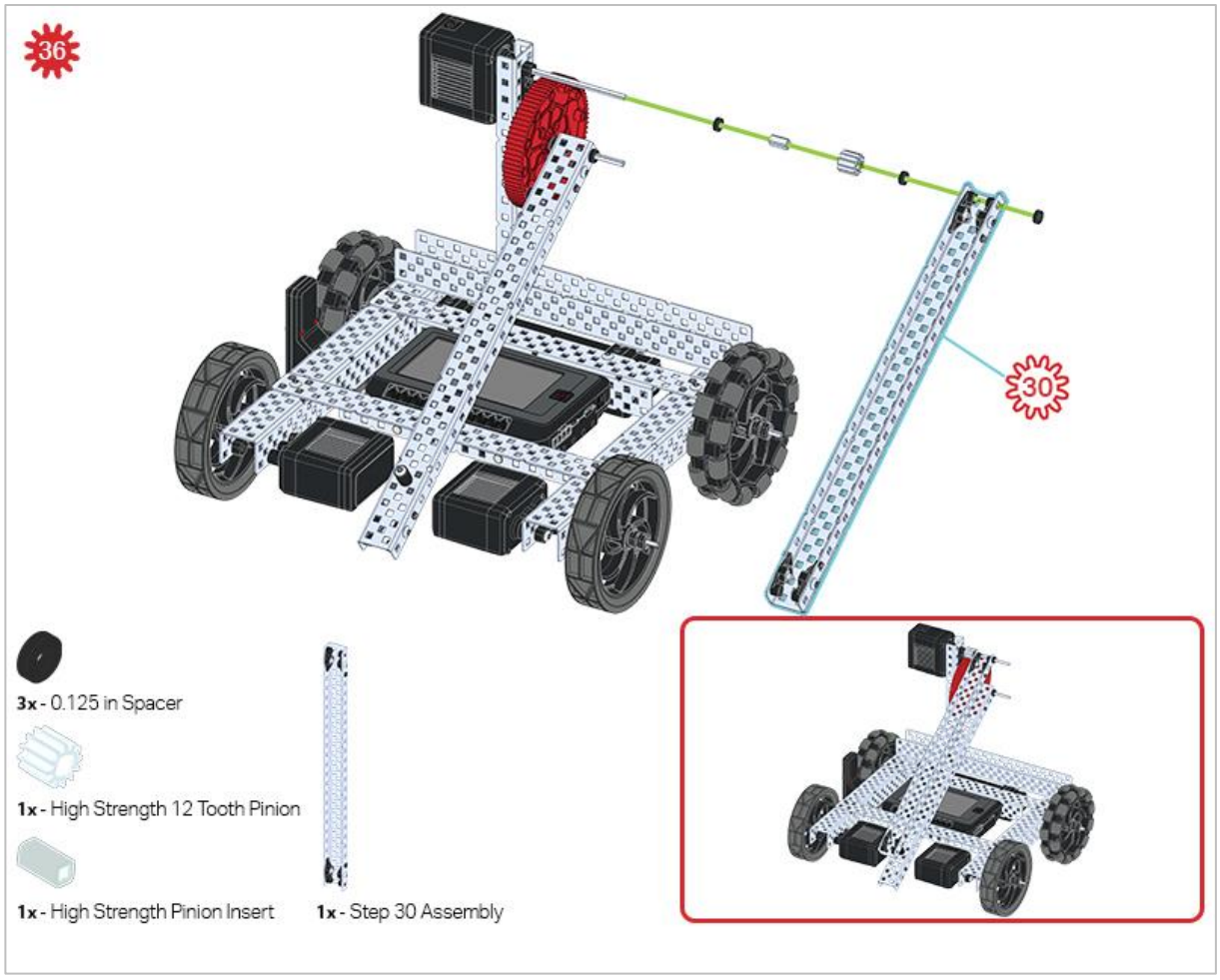


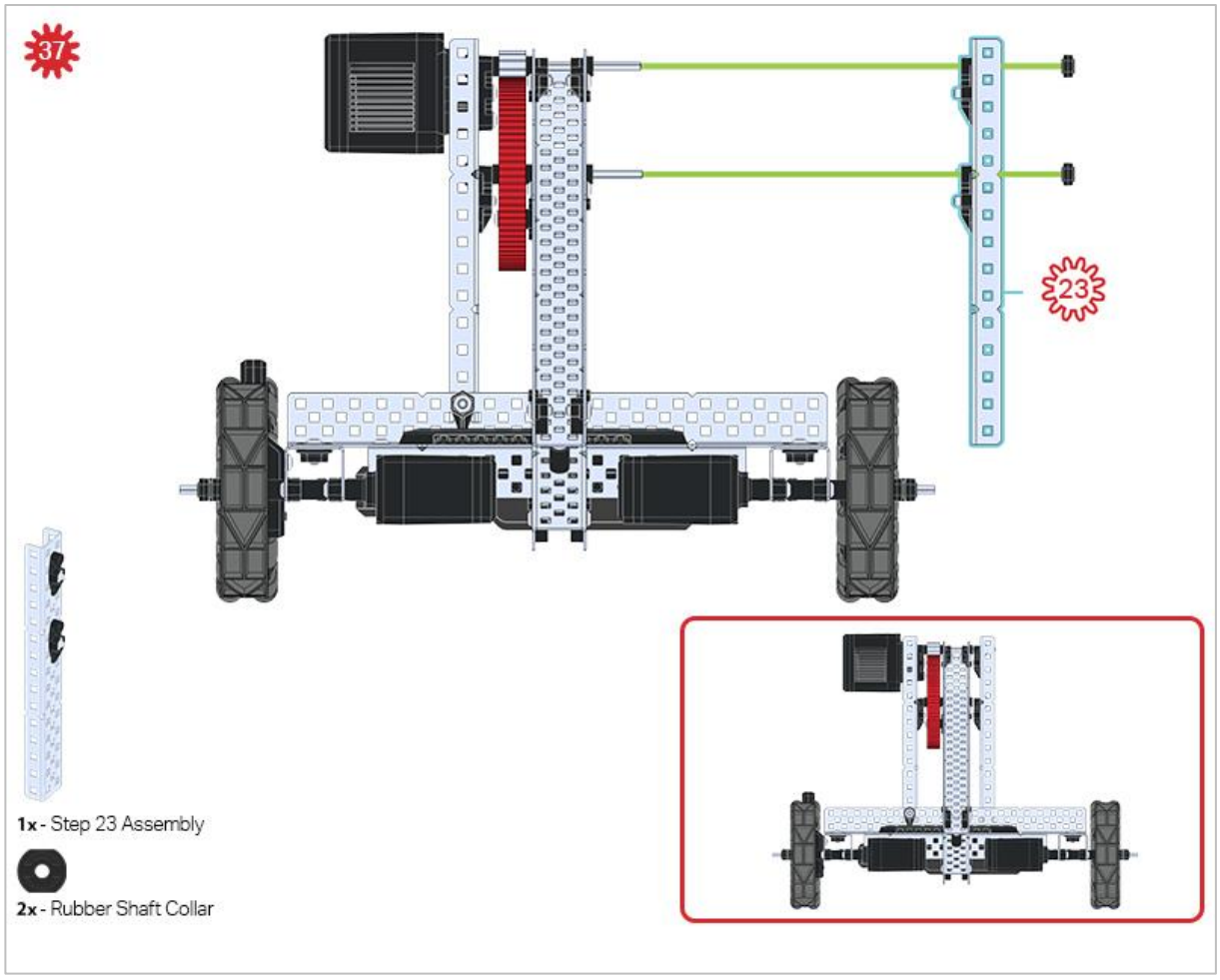
**1x** - 3.5 in Shaft




34
32
1x - Step 32 Assembly
2x - 0.125 in Spacer
1x - 8-32 x 1 in Screw


35
28
1x - Rubber Shaft Collar
2x - 0.125 in Spacer
1x - Step 28 Assembly

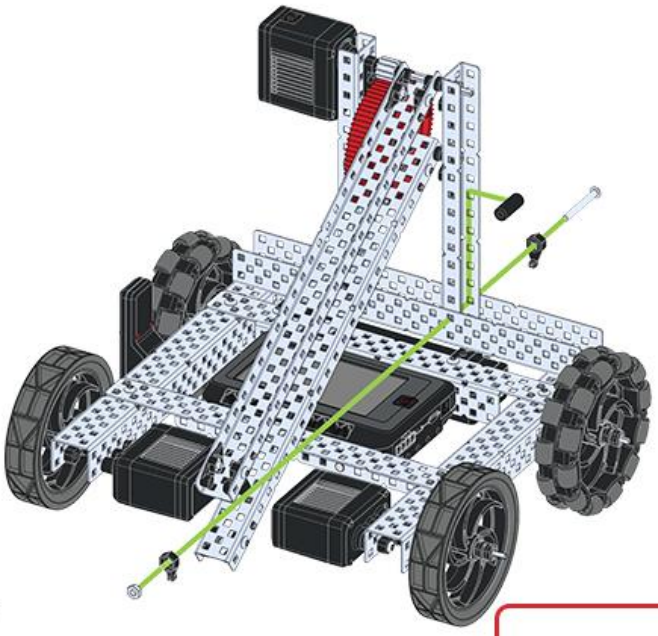

36
30
3x - 0.125 in Spacer
1x - High Strength 12 Tooth Pinion
1x - High Strength Pinion Insert
1x - Step 30 Assembly




**1x** - Step 23 Assembly

**2x** - Rubber Shaft Collar

38



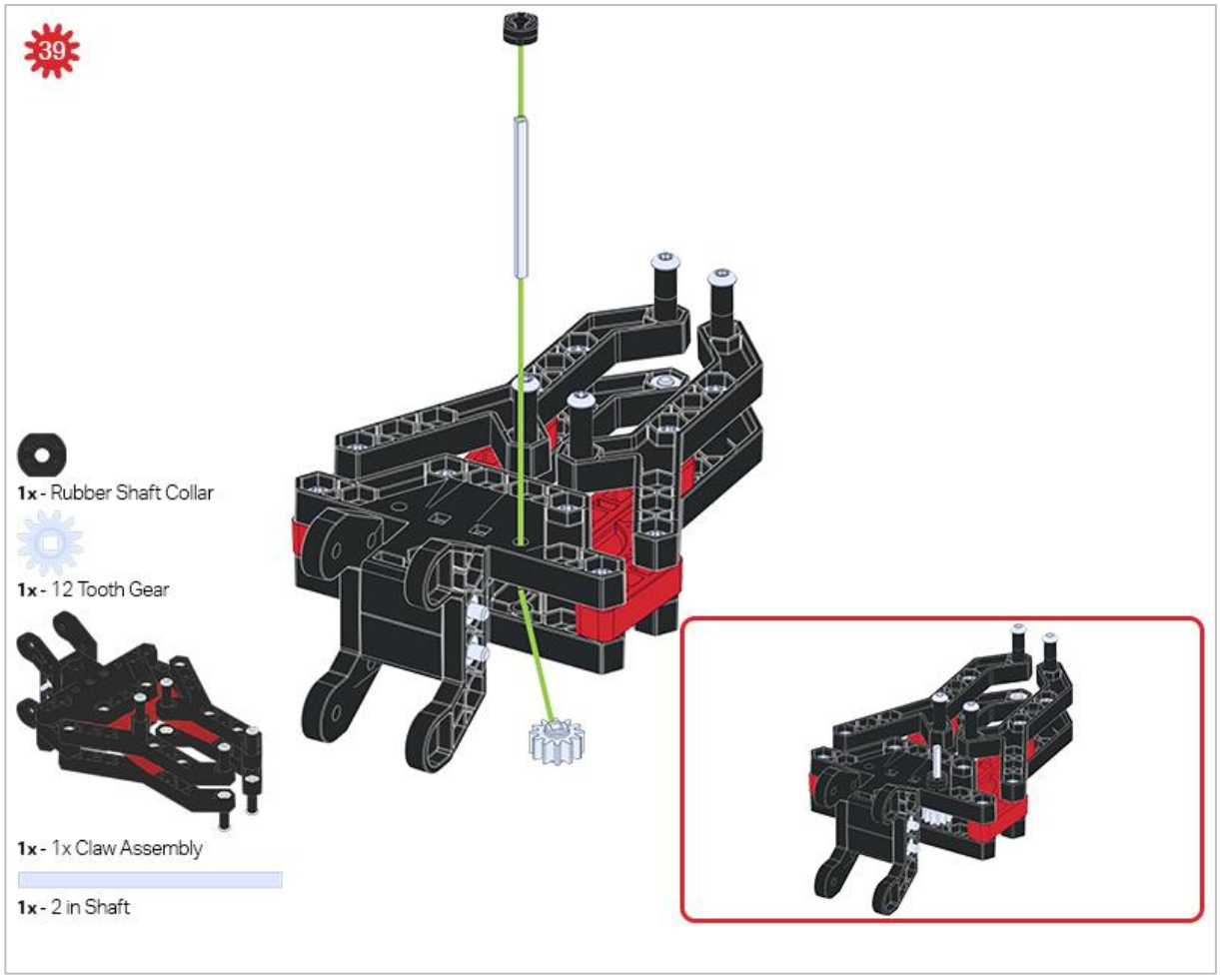
**1x** - 0.875 in Spacer



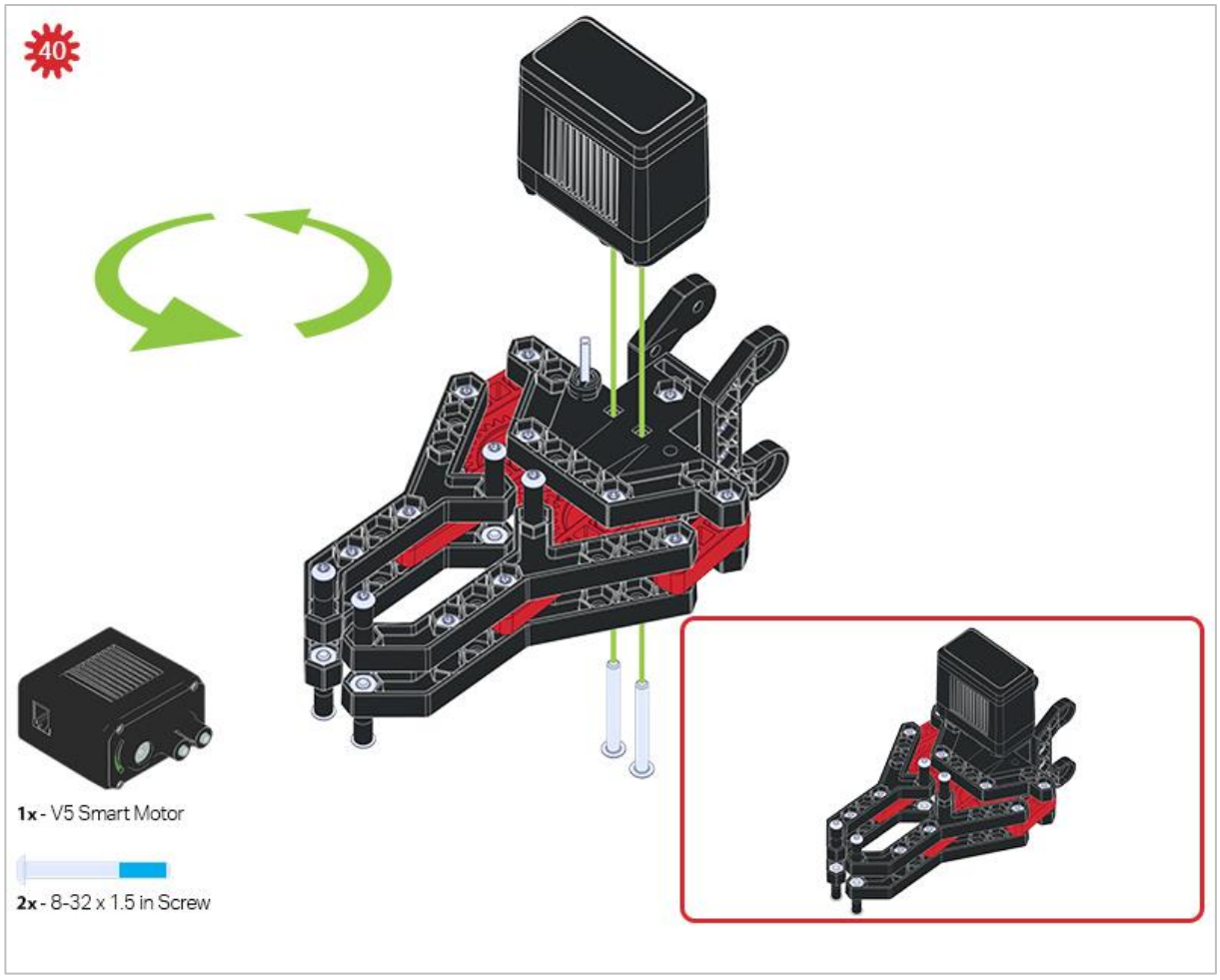
**1x** - 8-32 Nut

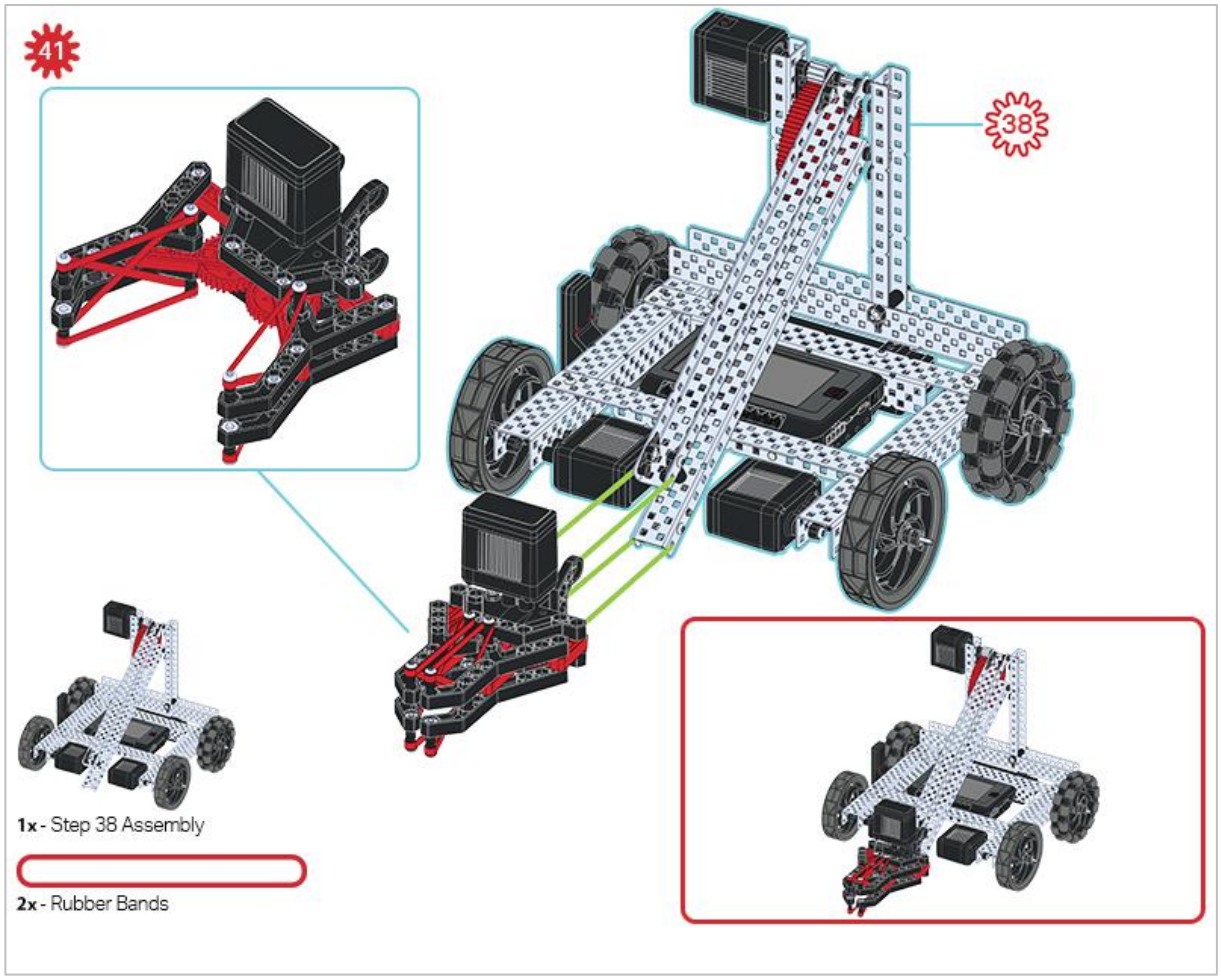
**2x** - 1 Post Hex Nut Retainer

**1x** - 8-32 x 1.5 in Screw



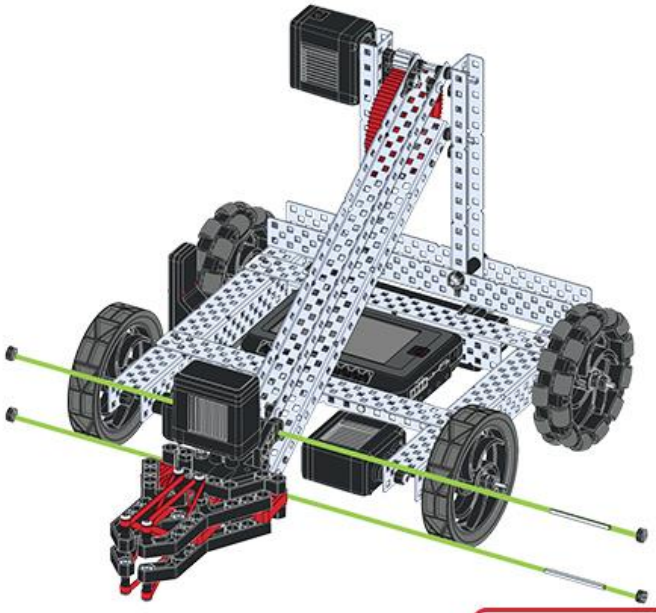

*Make sure the 12- tooth gear is installed on the right side of the claw.*


40
1x - V5 Smart Motor
2x - 8-32 x 1.5 in Screw



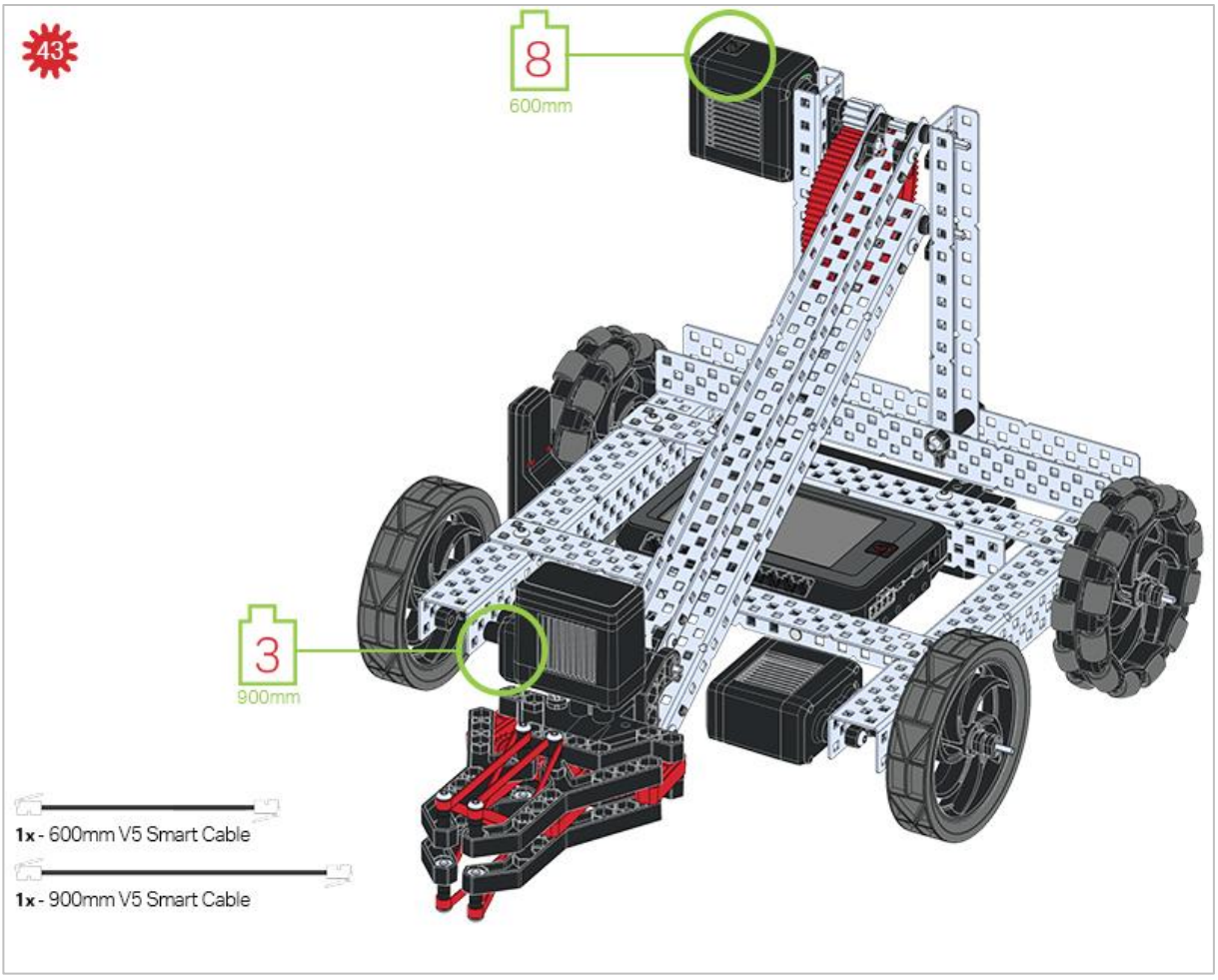

*Make sure that the port on the Smart Motor is facing the right side of the robot when the claw is installed (the same side as the V5 Radio).*

42



**4x** - Rubber Shaft Collar

**2x** - 2 in Shaft

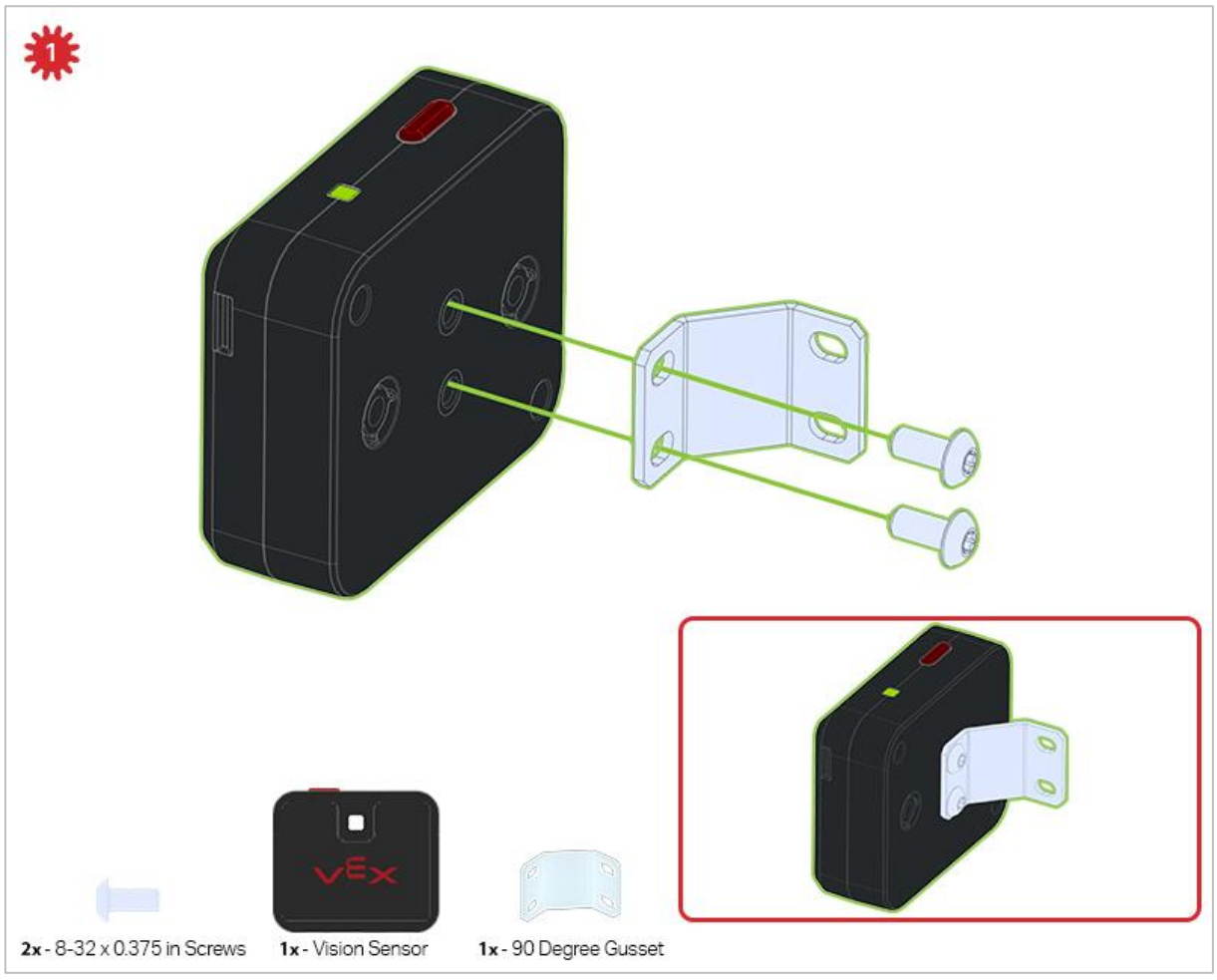


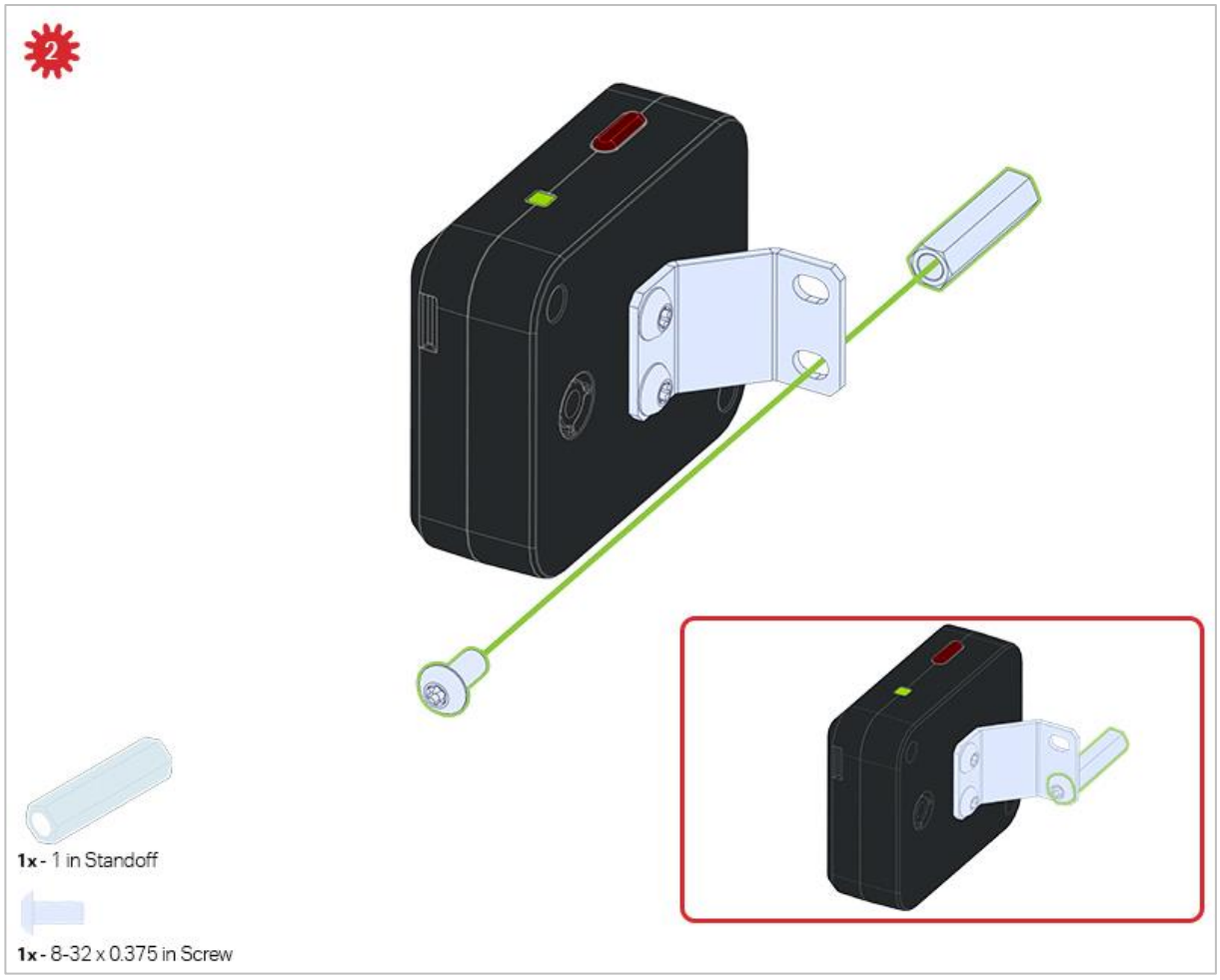

## Build Instruction Tips

Check the Appendix for information on how to use the new Hex Nut Retainers.

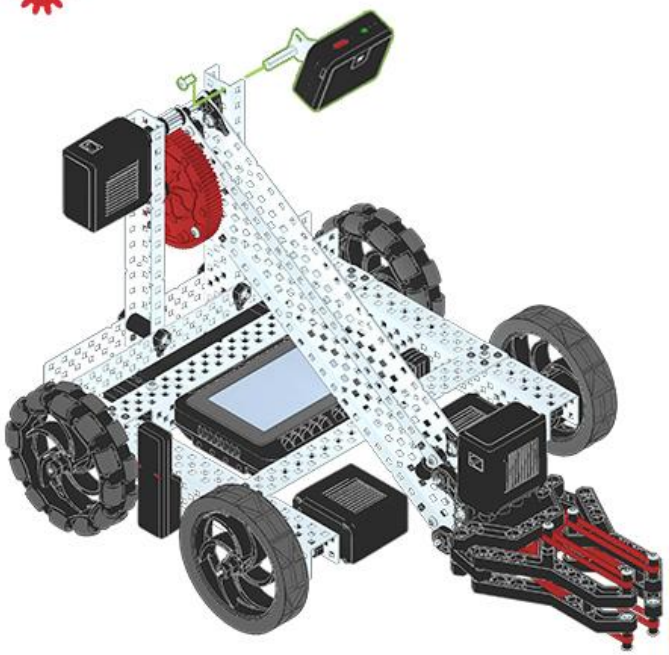
# Adding the Vision Sensor

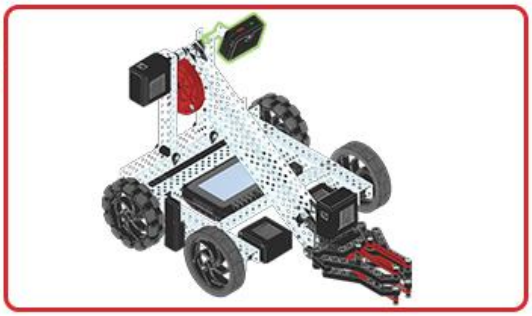
After the Clawbot has been assembled, use the building instructions below to add the Vision Sensor.




2
1x - 1 in Standoff
1x - 8-32 x 0.375 in Screw

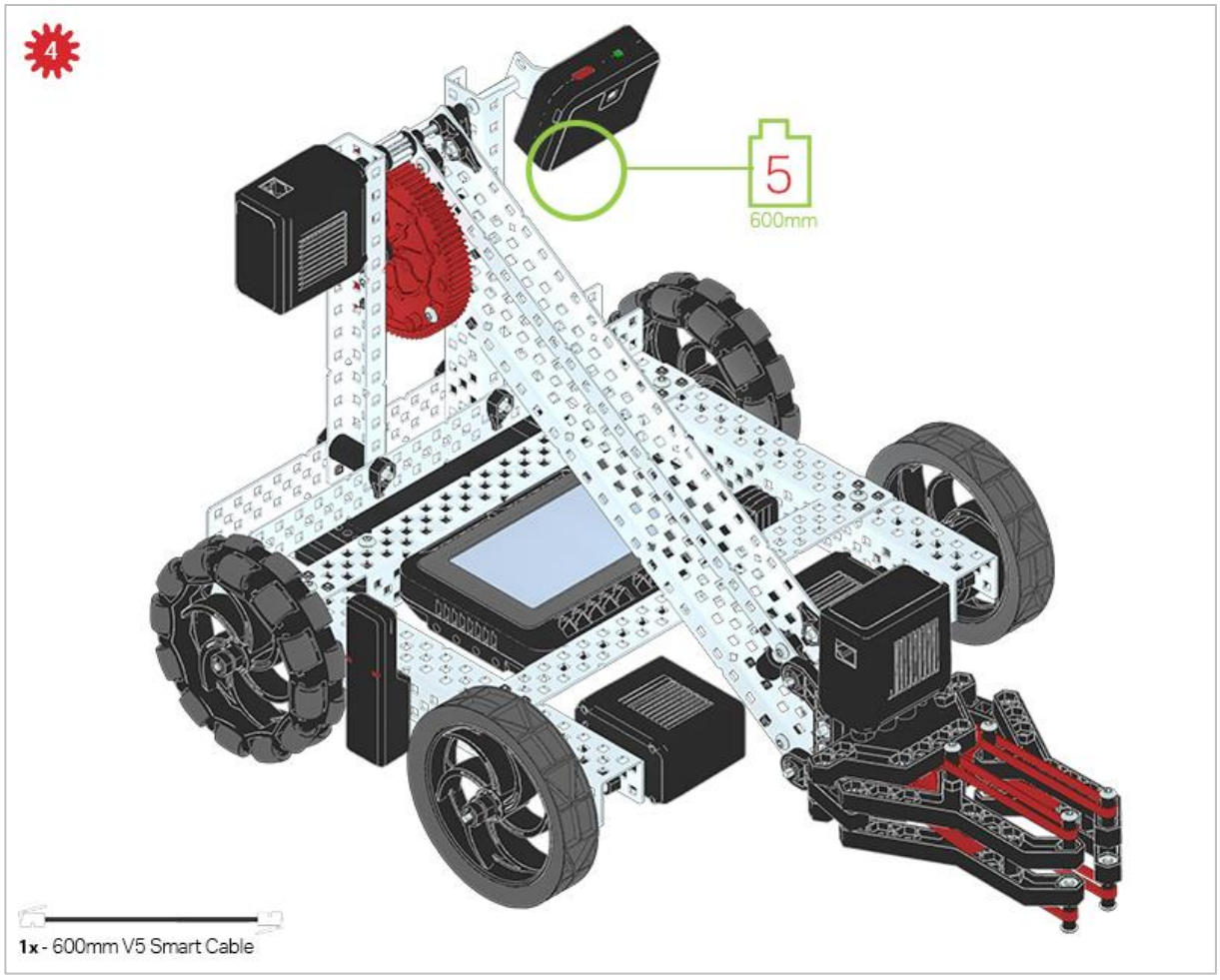
3







**1x** - 8-32 x 0.375 in Screw


4
5
600mm
1x - 600mm V5 Smart Cable

# Exploration

Now that you've finished the build, test what it does. Explore your build and then answer these questions in your engineering notebook.

Think about the placement of the Vision Sensor on the Clawbot:

- Why is the placement an important factor for this build?
    - What would happen if the Vision Sensor was on the back of the Clawbot?
    - What if the Vision Sensor was on the Arm?

SPARK
PLAY

Test your build, observe how it functions,
and fuel your logic and reasoning skills
through imaginative, creative play.

# What is a Vision Sensor? - VEXcode V5 Blocks

## Description

The Vision Sensor allows your robot to collect visual data from a live feed. A live feed is a streaming transmission of what a video camera is capturing. The Vision Sensor is like a smart camera that can observe, select, adjust, and store colors and objects that appear in its visual field.

*Vision Sensor 276-4850*

Capabilities:

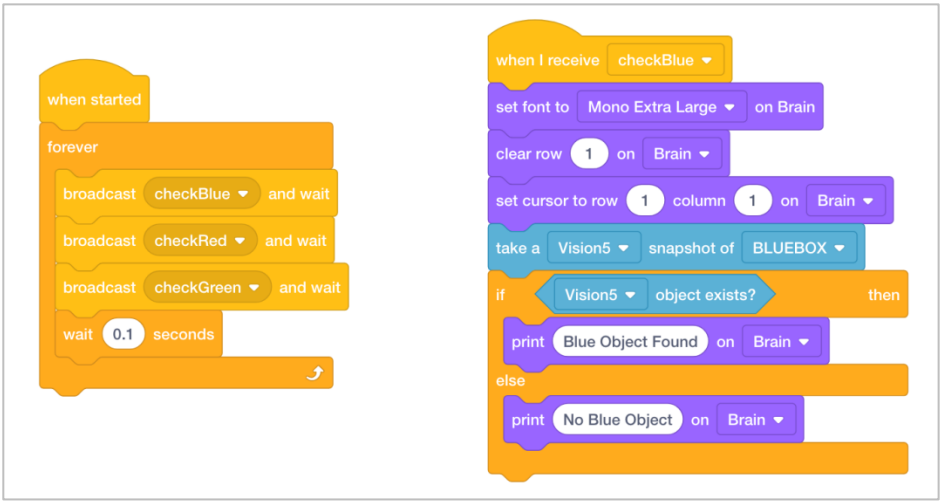
- This sensor can be used for recognizing colors and color patterns.
- This sensor can be used to follow an object.
- This sensor can be used to collect information about the environment.

The Vision Sensor allows the robot to use visual input data from its environment. The project can then determine how the visual input data should affect the robot's behavior. For example, the robot could perform actions (output) such as spinning motors or displaying results on the

LCD screen.

The Vision Sensor can also capture a snapshot of what is in front of it and analyze it according to what the user is asking. For example, a user can gather data from the snapshot such as, what color is the object? Is there an object detected at all? How large is the object (width and height)?

The robot can then make decisions based off of this data. The partial example project below shows how this is done. Three colors are being checked repeatedly after the project is started, and each color check is a different event. Only the event that checks for Blue is shown below. This stack has the robot print "Blue Object Found" if a blue object is detected or "No Blue Object" otherwise. The checkRed and checkGreen events not shown below have similar stacks for deciding what to print on the screen.



# Using the Vision Sensor - VEXcode V5 Blocks

Get the hardware required, your engineering notebook, and open VEXcode V5 Blocks.

**Hardware/Software Required:**

| Quantity | Hardware/Other Items |
|---|---|
| 1 | VEX V5 Classroom Starter Kit |
| 1 | VEXcode V5 Blocks (latest version, Windows, MacOS, Chromebook) |
| 1 | Engineering Notebook |
| 1 | Configuring a Vision Sensor (Tutorial) |
| 1 | Tuning the Vision Sensor (Tutorial) |
| 1 | Detecting Objects (Vision) example project |

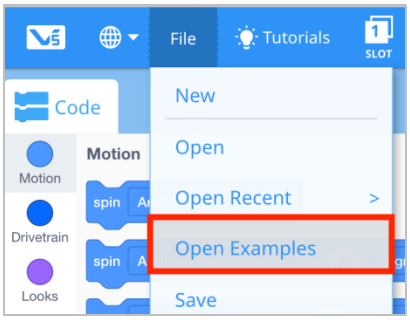
This activity will give you the tools to use the Vision Sensor.

You can use the Help information inside of VEXcode V5 Blocks to learn about the blocks. For guidance in using the Help feature, see the Using Help tutorial.


## 1. Open an Example Project.

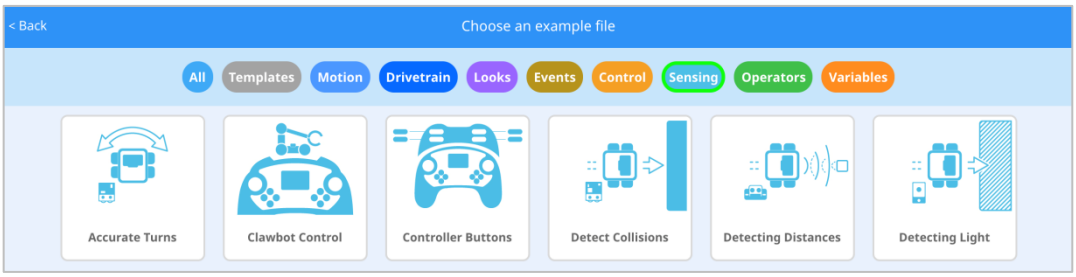
- VEXcode V5 Blocks contains many different example projects. You'll use one of them in this exploration. For help and tips on using example projects, check out the Using Examples and Templates tutorial.


Then, open the Detecting Objects (Vision) example project.



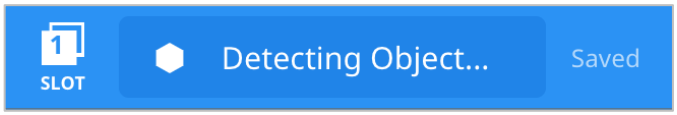

Complete the following steps:

- Open the File menu.
- Select **Open Examples**.
- Use the filter bar at the top of the application and select "Sensing."




- Select and open the **Detecting Objects (Vision)** example project.


- **Save** your project as Detecting Objects.

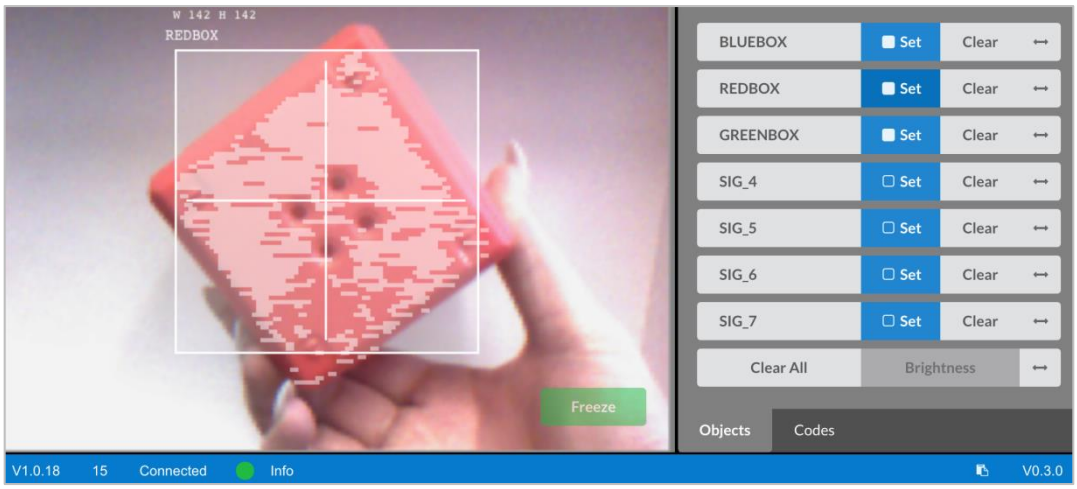



- Check to make sure the project name **Detecting Objects** is now in the window in the center of the toolbar.
- For addition help, view the Use Example Projects and Templates tutorial video.


## 2. Configuring and Using the Vision Sensor

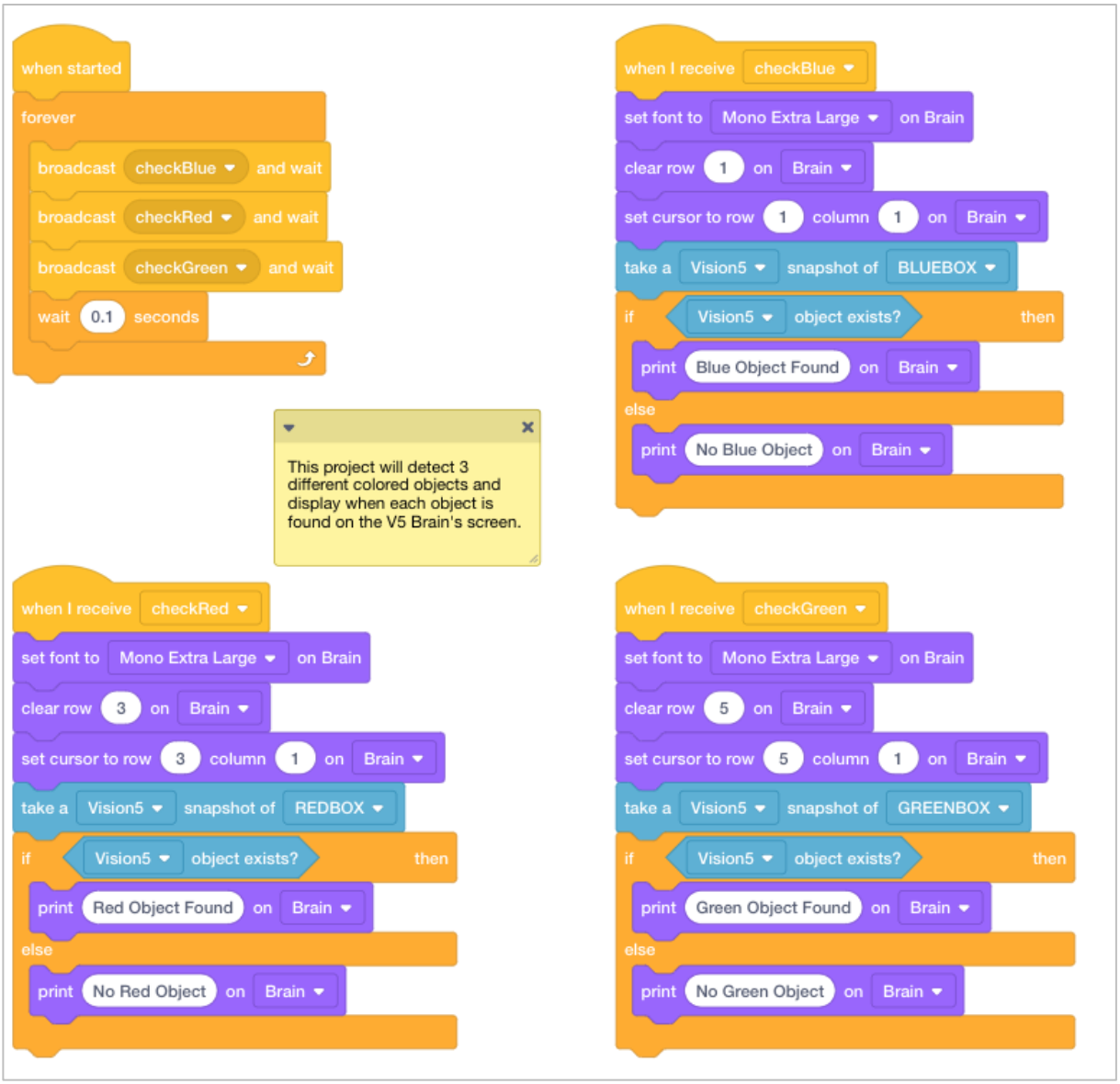
- Begin by watching the Configuring a Vision Sensor tutorial video.

- Next, configure the Vision Sensor for three colored objects: red, green, and blue.




- Open the previously saved Detecting Objects (Vision) example project.


- How is the Vision Sensor being used in this project? Predict what will happen when the project is run and write down the predictions in your engineering notebook.




- Download and Run the project. Place different colored objects in front of the Vision Sensor and observe the robot's behavior. Record in your engineering notebook how your prediction was different or correct compared to what you actually observed from the project.

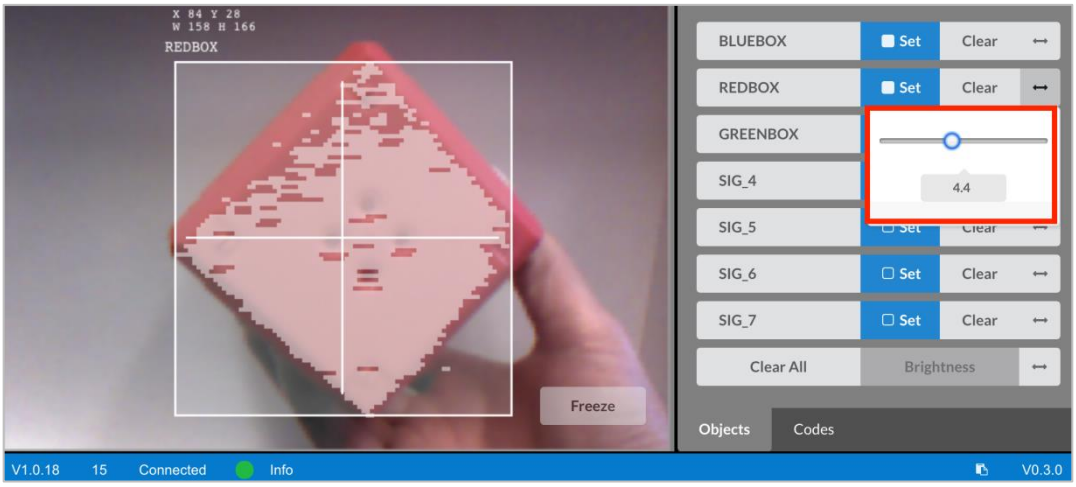
  For additional help, view the Download and Run a Project tutorial video.


## 3. Tuning the Vision Sensor

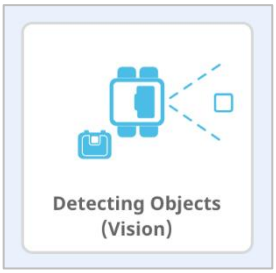
Often times an object is configured to be recognized by the Vision Sensor in one environment, for example, in a classroom. When the Vision Sensor is then taken into a different environment such as a competition setting, the object may not be recognized by the

Vision Sensor. This is often due to a change in lighting after the Vision Sensor has already been configured. To solve this problem, you may have to tune your Vision Sensor.

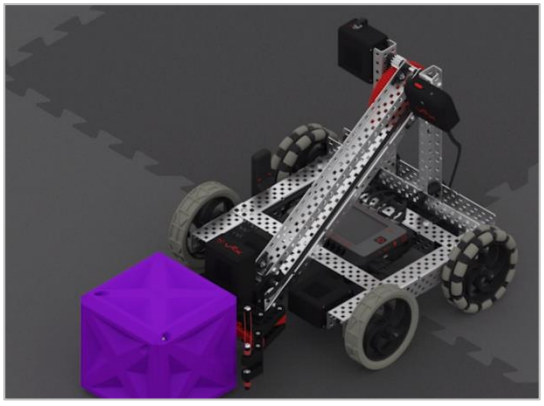
- Begin by watching the Tuning the Vision Sensor tutorial video.
- Next, Tune the Vision Sensor for the three colored objects: red, green, and blue.




- Open the previously saved Detecting Objects (Vision) example project.




- How will tuning the Vision Sensor affect how well it can detect objects? Take the Clawbot to a different part of the room with more or less light.



- Download and Run the project. Place different colored objects in front of the Vision Sensor and observe the robot's behavior. Document in your engineering notebook how well the Vision Sensor detects objects. Does the Vision Sensor need tuned after it changed locations?

  For additional help, view the Download and Run a Project tutorial video.


- Tune the Vision Sensor as necessary. Test the Vision Sensor after it has been tuned to determine if it can detect objects better and make adjustments as needed.

# What is a Vision Sensor? - VEXcode V5 Text

## Description

The Vision Sensor allows your robot to collect visual data from a live feed. A live feed is a streaming transmission of what a video camera is capturing. The Vision Sensor is like a smart camera that can observe, select, adjust, and store colors and objects that appear in its visual field.

*Vision Sensor 276-4850*

Capabilities:

- This sensor can be used for recognizing colors and color patterns.
- This sensor can be used to follow an object.
- This sensor can be used to collect information about the environment.

The Vision Sensor allows the robot to use visual input data from its environment. The project can then determine how the visual input data should affect the robot's behavior. For example, the robot could perform actions (output) such as spinning motors or displaying results on the

LCD screen.

The Vision Sensor can also capture a snapshot of what is in front of it and analyze it according to what the user is asking. For example, a user can gather data from the snapshot such as, what color is the object? Is there an object detected at all? How large is the object (width and height)?

The robot can then make decisions based off of this data. The partial example project below shows how this is done. Three colors are being checked repeatedly after the project is started, and each color check is a different event. Only the event that checks for Blue is shown below. This stack has the robot print "Blue Object Found" if a blue object is detected or "No Blue Object" otherwise. The checkRed and checkGreen events not shown below have similar stacks for deciding what to print on the screen.

```
#include "vex.h"

using namespace vex;

event checkBlue = event();
event checkRed = event();
event checkGreen = event();

void hasBlueCallback() {
  Brain.Screen.setFont(mono20);
  Brain.Screen.clearLine(1, black);
  Brain.Screen.setCursor(Brain.Screen.row(), 1);
  Brain.Screen.setCursor(1, 1);
  Vision5.takeSnapshot(Vision5__BLUEBOX);
  if (Vision5.objectCount > 0) {
    Brain.Screen.print("Blue Object Found");
  } else {
    Brain.Screen.print("No Blue Object");
  }
}

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();

  checkBlue(hasBlueCallback);

  while (true) {
    checkBlue.broadcast();
    checkRed.broadcast();
    checkGreen.broadcast();
    wait(1, seconds);
  }
}
```

# Using the Vision Sensor - VEXcode V5 Text

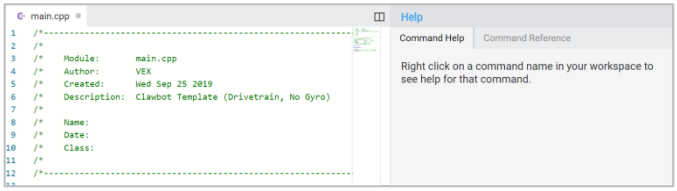
Get the hardware required, your engineering notebook, and open VEXcode V5 Text.

**Hardware/Software Required:**

| Quantity | Hardware/Other Items |
|---|---|
| 1 | VEX V5 Classroom Starter Kit |
| 1 | VEXcode V5 Text(latest version, Windows, MacOS) |
| 1 | Engineering Notebook |
| 1 | Configuring a Vision Sensor (Tutorial) |
| 1 | Tuning the Vision Sensor (Tutorial) |
| 1 | Detecting Objects (Vision) example project |

This activity will give you the tools to use the Vision Sensor.

You can use the Help information inside of VEXcode V5 Text to learn about the instructions.

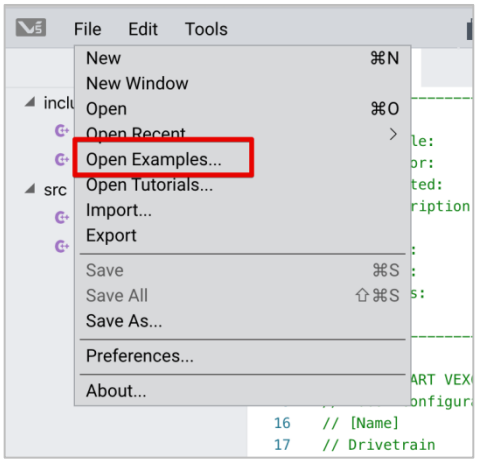



## 1. Open an Example Project.

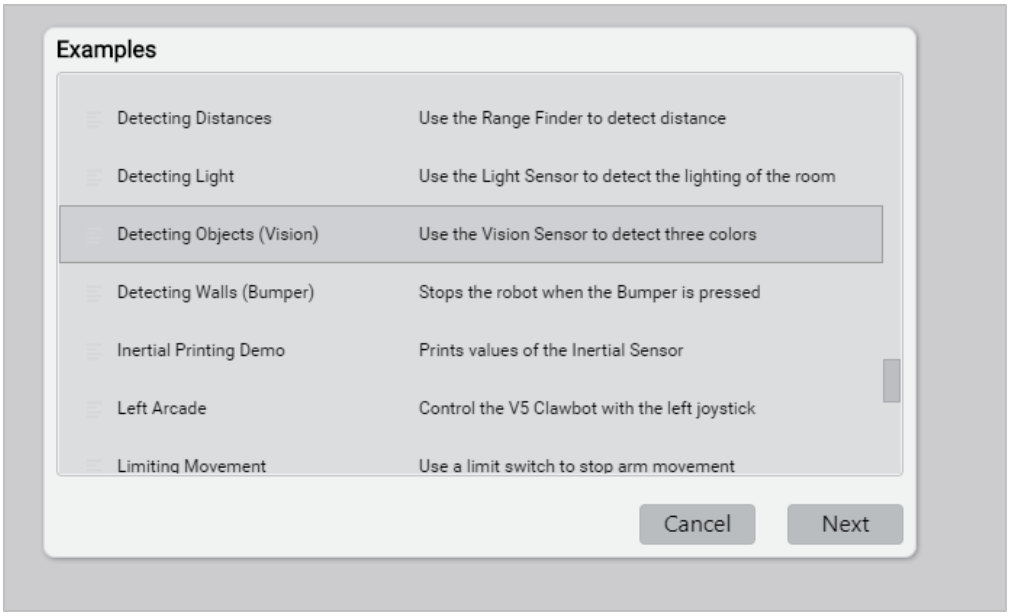
- VEXcode V5 Text contains many different example projects. You'll use one of them in this exploration. For help and tips on using example projects click here.

Complete the following steps:

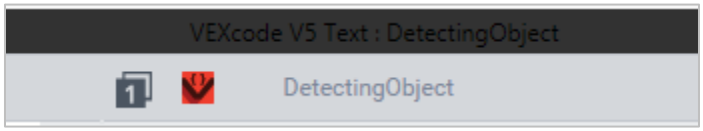
- Open the File menu.
- Select **Open Examples**.



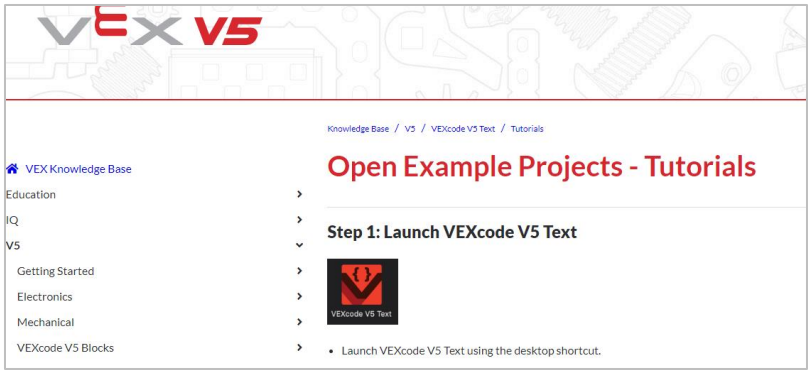

- Select and open the **Detecting Objects (Vision)** example project.




- **Save** your project as Detecting Objects.

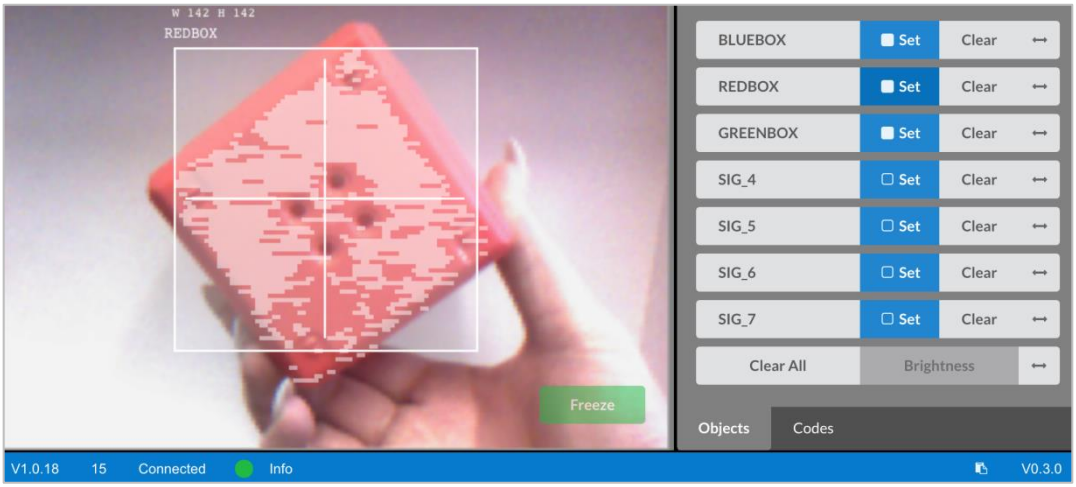



- Check to make sure the project name **Detecting Objects** is now in the window in the center of the toolbar.
- For addition help in opening an example project, click here.

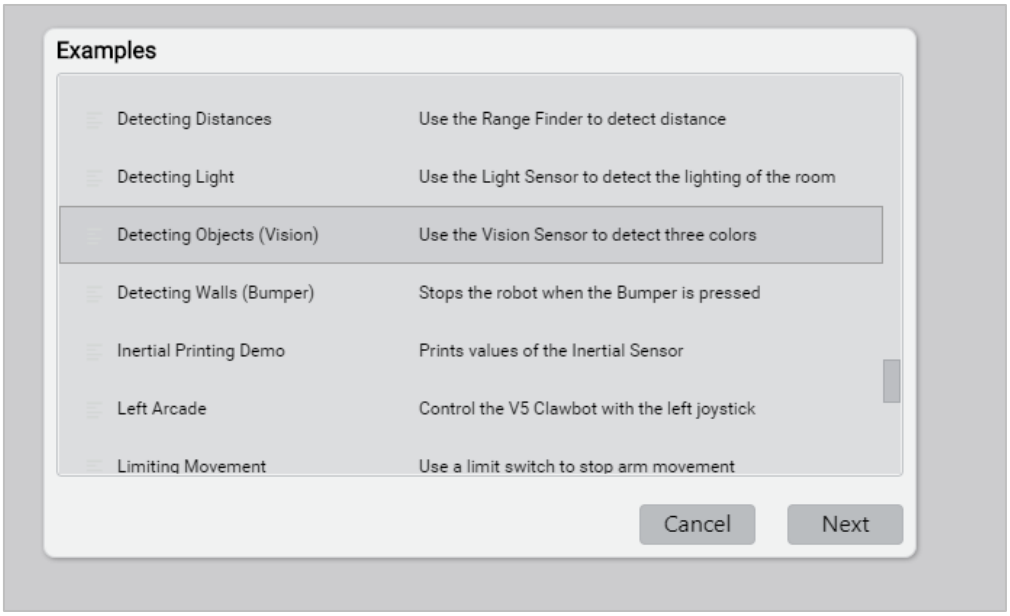



## 2. Configuring and Using the Vision Sensor

- Begin by reading Vision Sensor - Robot Config - VEXcode Text .
- Then,  read Vision Sensor Adjustments-Robot Config-VEXcode Text
- Next, configure the Vision Sensor for three colored objects: red, green, and blue.




- Open the previously saved Detecting Objects (Vision) example project.




- How is the Vision Sensor being used in this project? Predict what will happen when the project is run and write down the predictions in your engineering notebook.

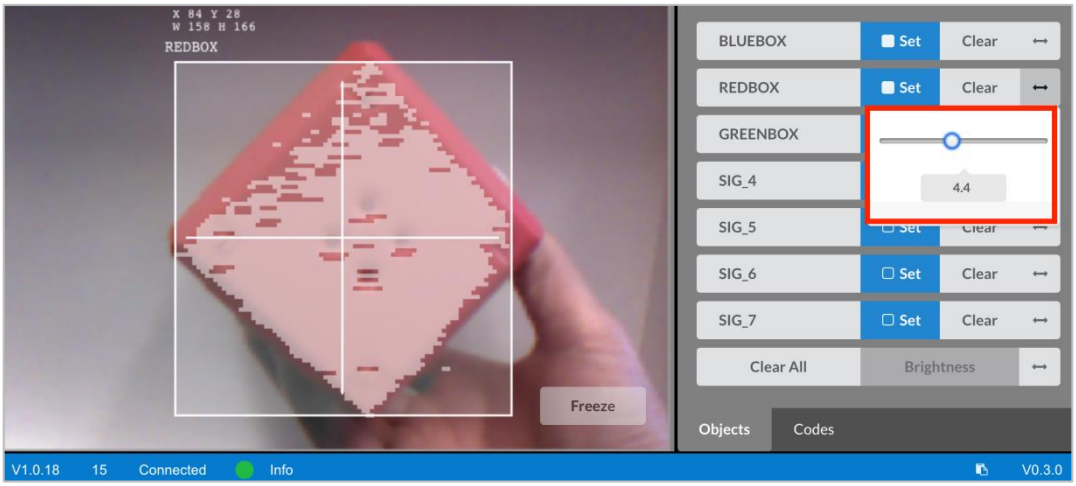
```cpp
#include "vex.h"

using namespace vex;

event checkRed = event();
event checkBlue = event();
event checkGreen = event();

void hasBlueCallback() {
  Brain.Screen.setFont(mono40);
  Brain.Screen.clearLine(1, black);
  Brain.Screen.setCursor(Brain.Screen.row(), 1);
  Brain.Screen.setCursor(1, 1);
  Vision5.takeSnapshot(Vision5__BLUEBOX);
  if (Vision5.objectCount > 0) {
    Brain.Screen.print("Blue Object Found");
  } else {
    Brain.Screen.print("No Blue Object");
  }
}

void hasRedCallback() {
  Brain.Screen.setFont(mono40);
  Brain.Screen.clearLine(3, black);
  Brain.Screen.setCursor(Brain.Screen.row(), 1);
  Brain.Screen.setCursor(3, 1);
  Vision5.takeSnapshot(Vision5__REDBOX);
  if (Vision5.objectCount > 0) {
    Brain.Screen.print("Red Object Found");
  } else {
    Brain.Screen.print("No Red Object");
  }
}

void hasGreenCallback() {
  Brain.Screen.setFont(mono40);
  Brain.Screen.clearLine(5, black);
  Brain.Screen.setCursor(Brain.Screen.row(), 1);
  Brain.Screen.setCursor(5, 1);
  Vision5.takeSnapshot(Vision5__GREENBOX);
  if (Vision5.objectCount > 0) {
    Brain.Screen.print("Green Object Found");
  } else {
    Brain.Screen.print("No Green Object");
  }
}

int main() {
  // Initializing Robot Configuration. DO NOT REMOVE!
  vexcodeInit();

  checkBlue(hasBlueCallback);
  checkRed(hasRedCallback);
  checkGreen(hasGreenCallback);
```

- Download and Run the project. Place different colored objects in front of the Vision Sensor and observe the robot's behavior. Record in your engineering notebook how your prediction was different or correct compared to what you actually observed from the project.

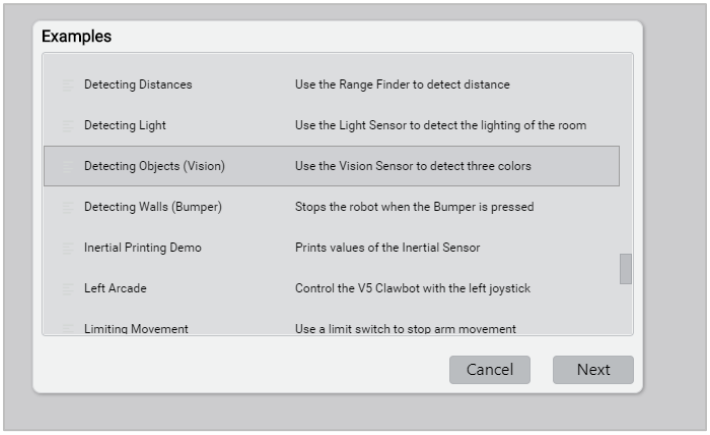
## 3. Tuning the Vision Sensor

Often times an object is configured to be recognized by the Vision Sensor in one environment, for example, in a classroom. When the Vision Sensor is then taken into a different environment such as a competition setting, the object may not be recognized by the Vision Sensor. This is often due to a change in lighting after the Vision Sensor has already been configured. To solve this problem, you may have to tune your Vision Sensor.

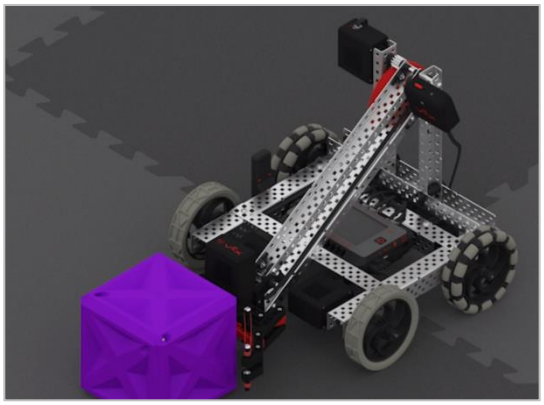
- Begin by watching the Tuning the Vision Sensor tutorial video.
- Next, adjust the Vision Sensor for the three colored objects: red, green, and blue.




- Open the previously saved Detecting Objects (Vision) example project.




- How will tuning the Vision Sensor affect how well it can detect objects? Take the Clawbot to a different part of the room with more or less light.



- Download and Run the project. Place different colored objects in front of the Vision Sensor and observe the robot's behavior. Document in your engineering notebook how well the Vision Sensor detects objects. Does the Vision Sensor need tuned after it changed locations?
- Tune the Vision Sensor as necessary. Test the Vision Sensor after it has been tuned to determine if it can detect objects better and make adjustments as needed.


Become a 21st century problem solver
by applying the core skills and concepts
you learned to other problems.

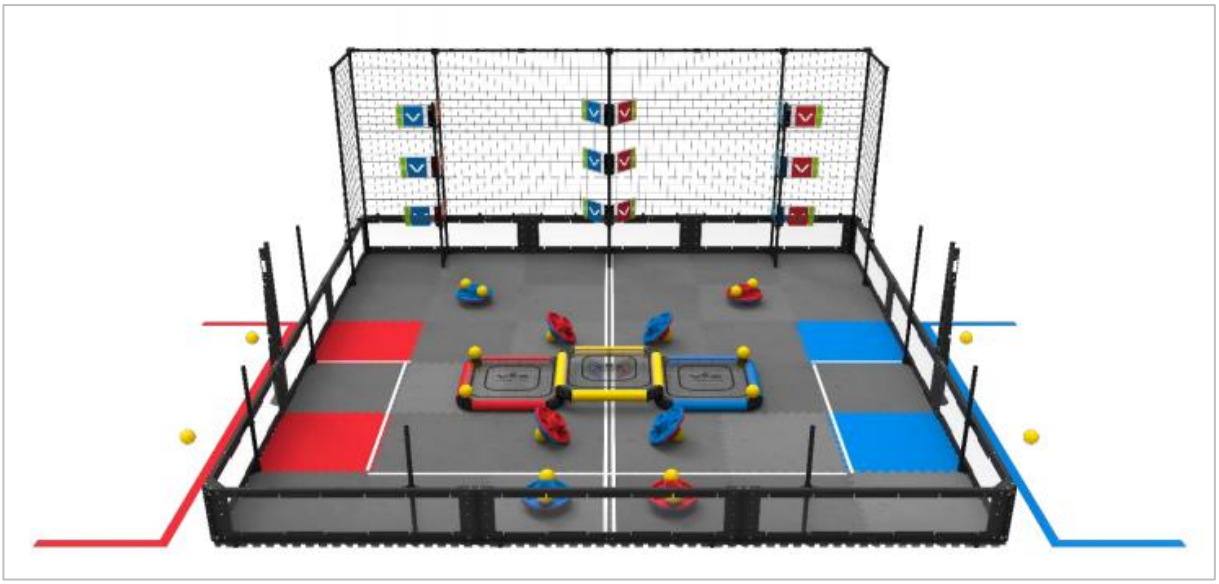
# Perception with Self-Driving Vehicles

*A self-driving vehicle with an array of sensors*

## Perception with Self-Driving Vehicles

Self-driving vehicles use a wide array of sensors to perceive their surroundings, on-board computers to combine and process the data from all of those sensors, and one or more motors to safely move the vehicle along the road using AI (artificial intelligence). Self-driving vehicles often make use of high resolution cameras to be able to recognize things such as pedestrians, road signs, and other vehicles. Vision data from the cameras is often combined with sensors that use lasers to measure distance between the vehicle and other objects. This allows the on-board computers to make decisions based on what types of objects are in their environment, and how far away each object is.

# Competition Connection: Turning Point - VEXcode V5 Blocks



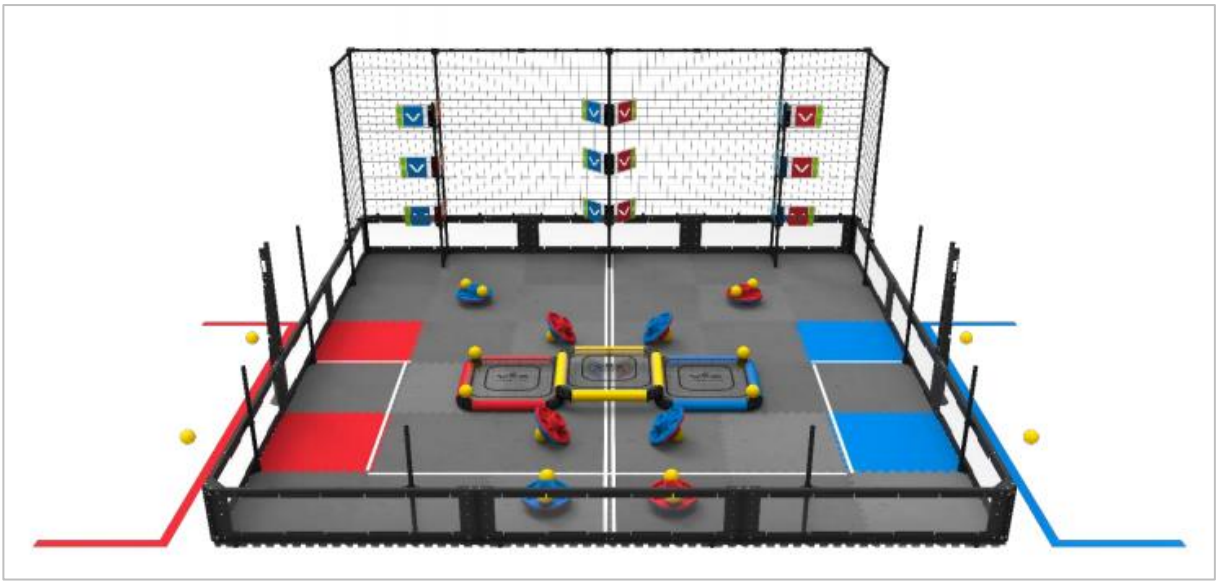
*VRC 2018-2019 Turning Point Field*

## Robot Capabilities

The 2018 - 2019 VEX Robotics Competition game Turning Point required players to toggle flags among other game elements. There were nine flags total: three bottom flags that could be toggled by the robot, and the six high flags that could only be toggled by hitting them with competition ball game pieces. Competition teams needed to come up with a way to hit the higher flags using a ball launcher. If you can imagine, programming the robot to hit the flags using ball game pieces by measuring may not always be accurate. If the robot makes one wrong turn during the autonomous period, there is a possibility that none of the flags would be hit because the calculations would be off. Similarly, for the Driving Skills challenge, it may be hard for teams to manually line up the robot enough to launch the ball properly.

VEX Robotics Competition games often use different colored game elements and so, a great advantage would be to design a robot with a Vision Sensor. In the Turning Point game, for example, a Vision Sensor could have been used to detect flags and then align the robot properly in order to make accurate shots. It is important to note that Vision Sensors are sensitive to the lighting in different environments. In competition settings, there is time to tune the Vision Sensor and tuning the Vision Sensor is an important practice that teams should adopt.

# Competition Connection: Turning Point - VEXcode V5 Text



*VRC 2018-2019 Turning Point Field*

## Robot Capabilities

The 2018 - 2019 VEX Robotics Competition game Turning Point required players to toggle flags among other game elements. There were nine flags total: three bottom flags that could be toggled by the robot, and the six high flags that could only be toggled by hitting them with competition ball game pieces. Competition teams needed to come up with a way to hit the higher flags using a ball launcher. If you can imagine, programming the robot to hit the flags using ball game pieces by measuring may not always be accurate. If the robot makes one wrong turn during the autonomous period, there is a possibility that none of the flags would be hit because the calculations would be off. Similarly, for the Driving Skills challenge, it may be hard for teams to manually line up the robot enough to launch the ball properly. Thus, skilled teams would program the robot using the Vision Sensor to detect flags and then align the robot properly in order to make accurate shots.


Is there a more efficient way to come to the same conclusion? Take what you've learned and try to improve it.

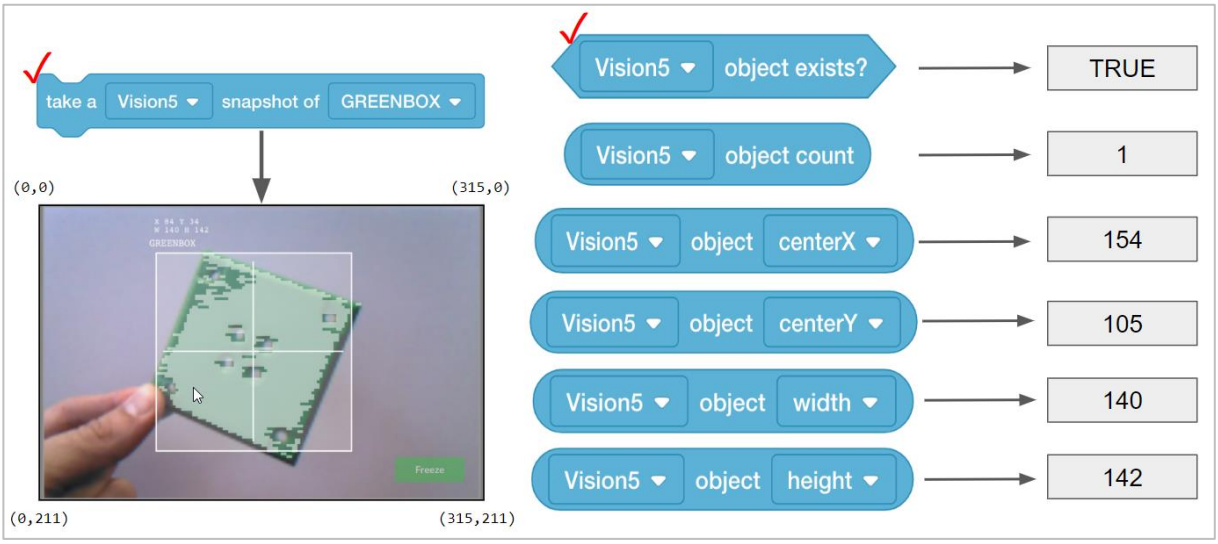
# Prepare for the Vision Data Challenge - VEXcode V5 Blocks

## The Vision Sensor's Sensing Blocks

VEXcode V5 Blocks has Sensing blocks for the Vision Sensor. The first two you already used in the Play section to take a snapshot and to check if the object exists.

In the figure below, you see that the *snapshot* block captured the GREENBOX snapshot. The object, GREENBOX, was identified in the snapshot and so the answer of whether it exists is TRUE.

Let's look at these other Sensing blocks and what their values tell us.




- The *object count* block tells us how many GREENBOX objects are in the snapshot. Here, there is only 1 detected.
- The center X value tells us whether the GREENBOX object is to the left or right of the robot's center point. Remember, the Vision Sensor should be mounted in the middle of the robot facing forward and so the snapshot's view is the robot's view.
  - If center X is greater than 157.5, the object is to the right of the robot's center point.
  - If center X is less than 157.5, the object is to the left of the robot's center point.
- The center Y value tells us whether the GREENBOX is higher or lower than the robot's center point.

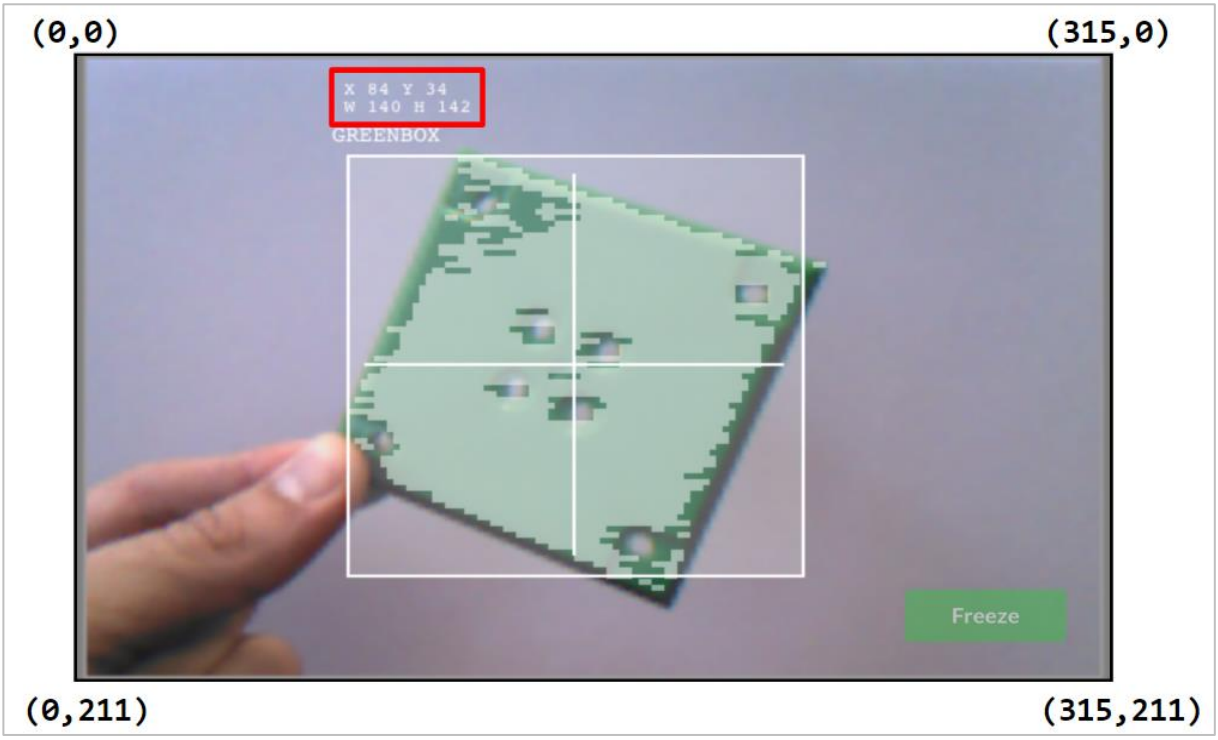
- If center Y is greater than 105.5, the object is lower than the robot's center point.
- If center Y is less than 105.5, the object is higher than the robot's center point.

- The width and height values tell us how close the GREENBOX is to the robot.
  - The same-sized object will be larger in width and height as it gets closer to the robot.

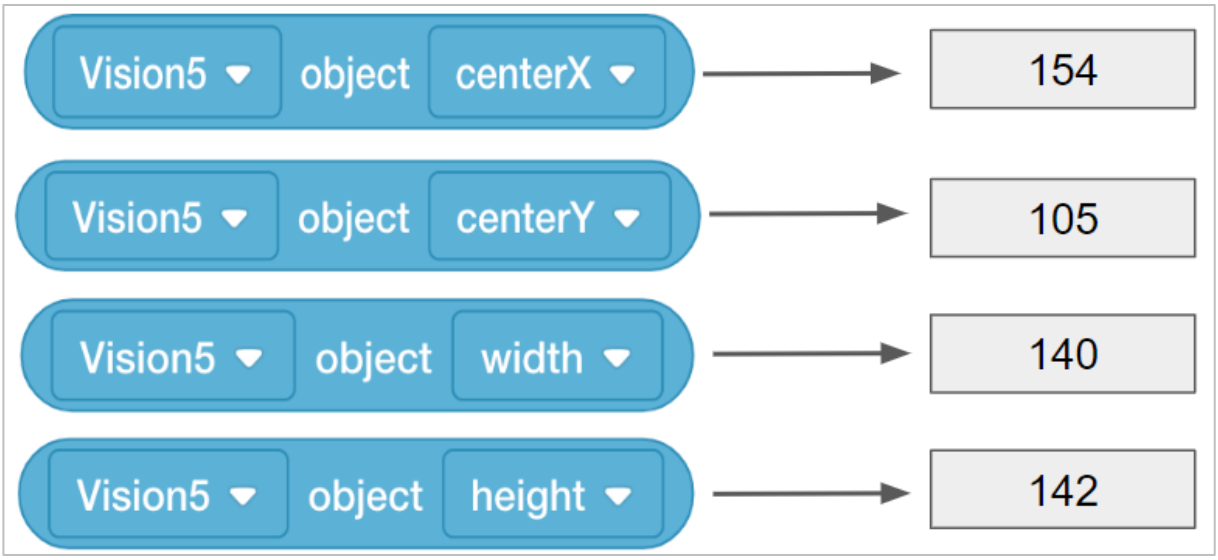
## How are the center X and center Y values calculated?

The values are calculated based on the coordinates within the snapshot. The width and height of the object are already calculated.

The Vision Sensor tracks the X and Y values of the upper left corner of the object. Below, those coordinates are (84, 34).

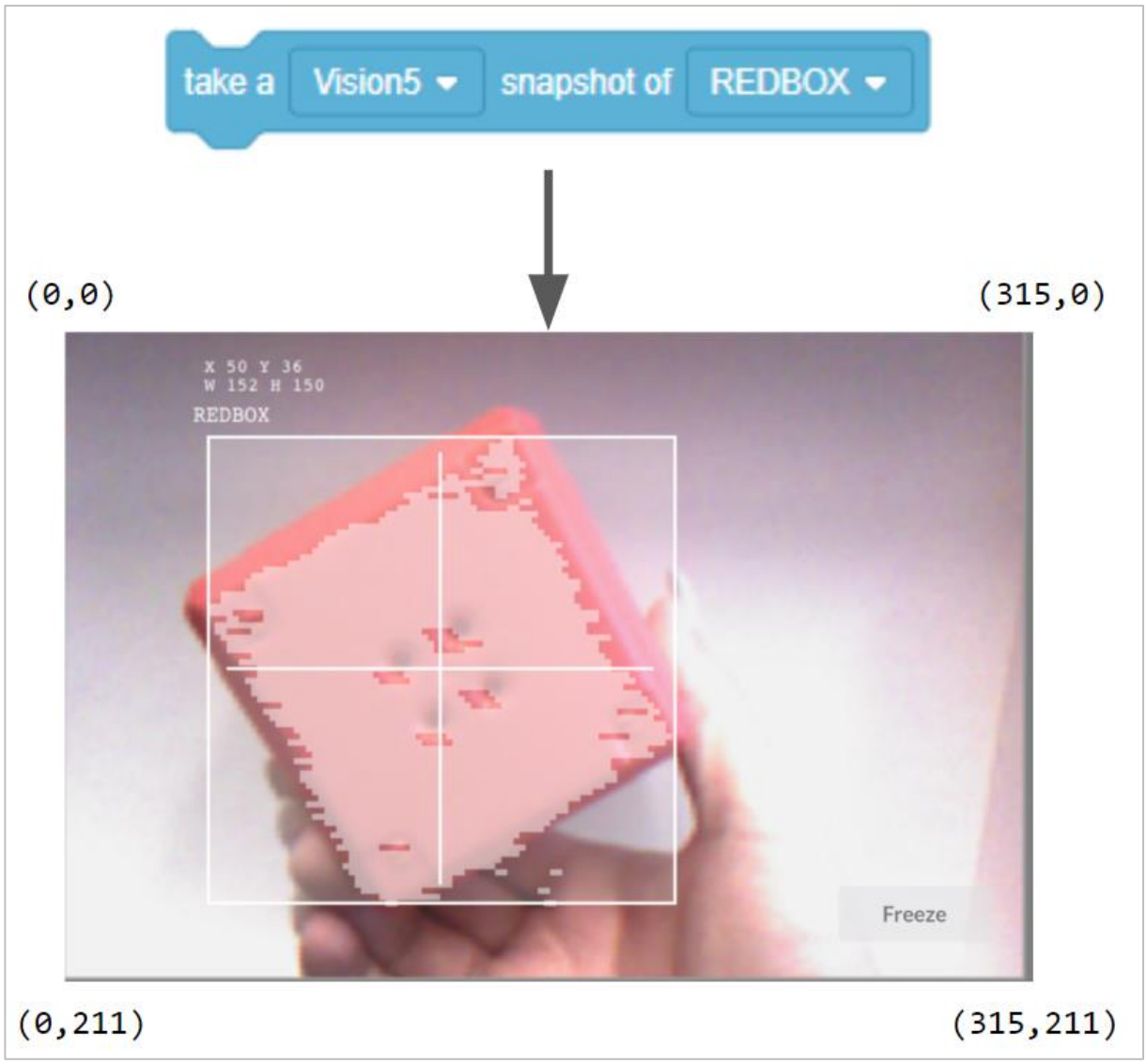



The center X and center Y values can be calculated based off of the coordinates of the upper left corner (84, 34), and the width (W 140) and height (H 142) values provided.




- centerX = 140/2 + 84 = **154**
  - centerX = half the width of the object added to its leftmost X coordinate
- centerY = 142/2 + 34 = **105**
  - centerY = half the height of the object added to its topmost Y coordinate

# Practice for the Vision Data Challenge - VEXcode V5 Blocks

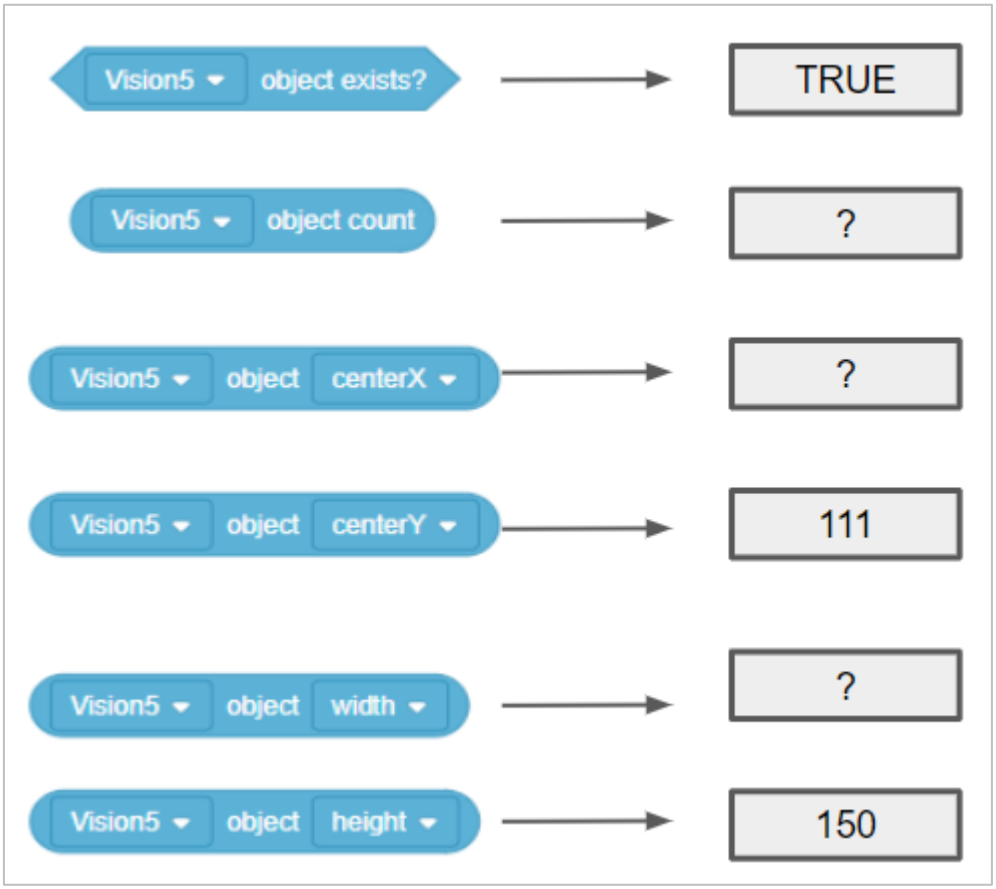



## Add the missing values below in your engineering notebook.

Here are the provided data from the snapshot:

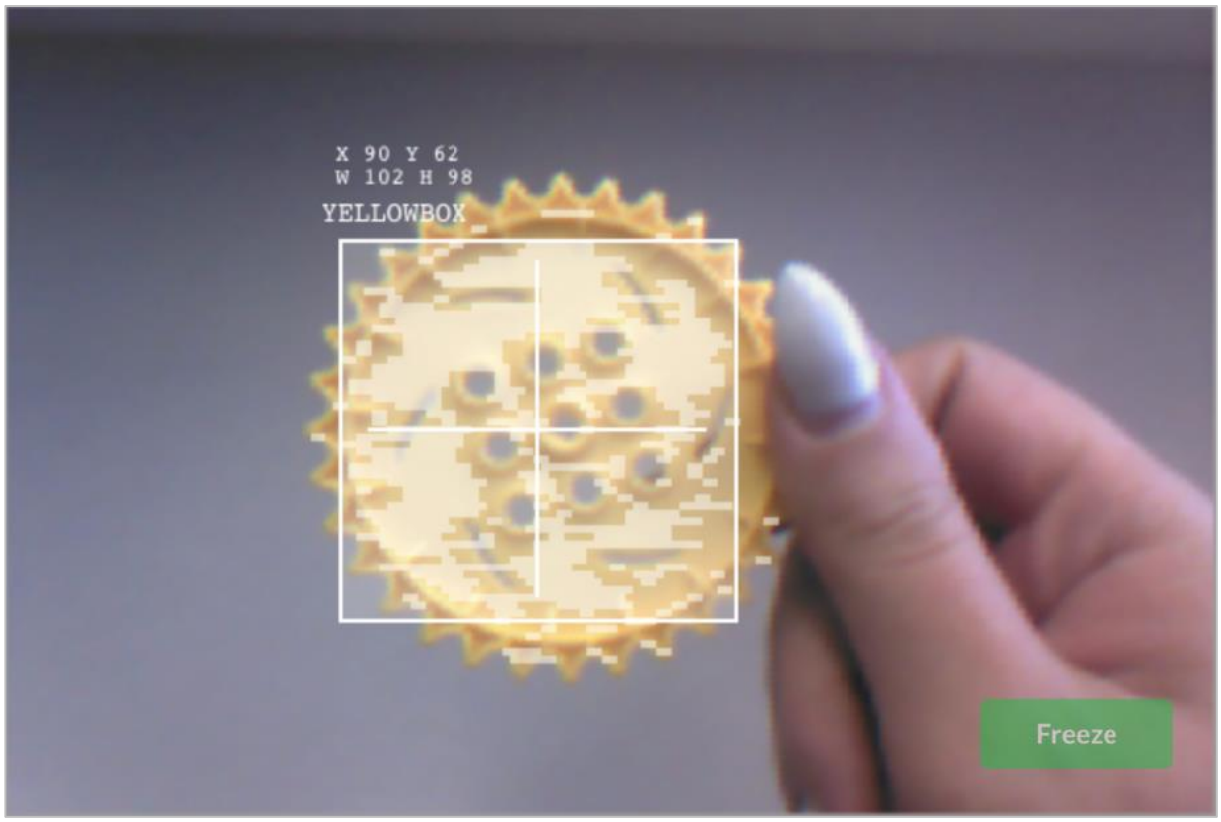
- X = 50
- Y = 36
- W = 152

- H = 150



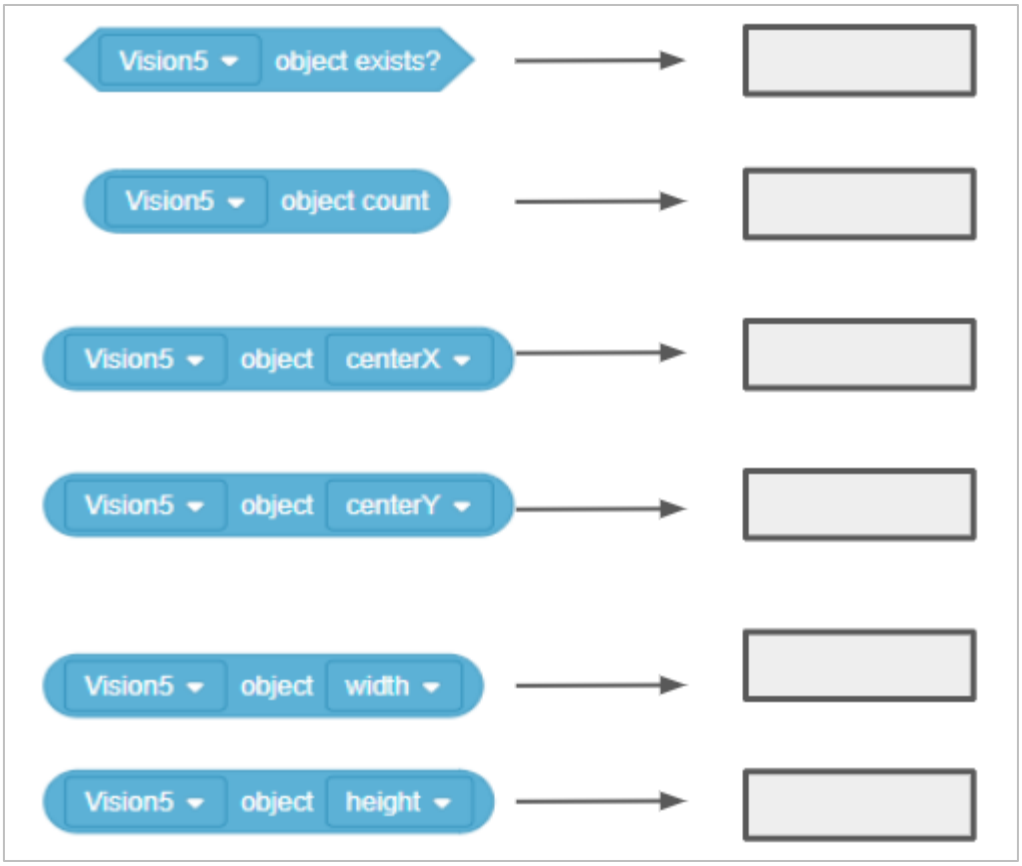

- Is the REDBOX to the left or to the right of the robot's center point?
- Is the REDBOX higher or lower than the robot's center point?

# The Vision Data Challenge - VEXcode V5 Blocks




## Complete the Vision Data Challenge by answering the questions and filling in the missing data in your engineering notebook.

- Which of these blocksinstructions was used to take the snapshot above?
  - 
  - 
- Fill in these values:




- Is YELLOWBOX to the left or to the right of the robot's center point?
- Is YELLOWBOX above or below the robot's center point?
- YELLOWBOX is **NOT** the best name to give this object if you want to easily recognize which color signature is which. Which of these is a better name? Why?
    - YELLOWGEAR
    - YELLOWCUBE

# Prepare for the Vision Data Challenge - VEXcode V5 Text

## The Vision Sensor's Sensing Instructions

VEXcode V5 has Sensing instruction for the Vision Sensor. The first two you already used in the Play section to take a snapshot and to check if the object exists.

In the figure below, you see that the *snapshot* captured the GREENBOX snapshot. The object, GREENBOX, was identified in the snapshot and so the answer of whether it exists is TRUE.

Let's look at these other Sensing instructions and what their values tell us.

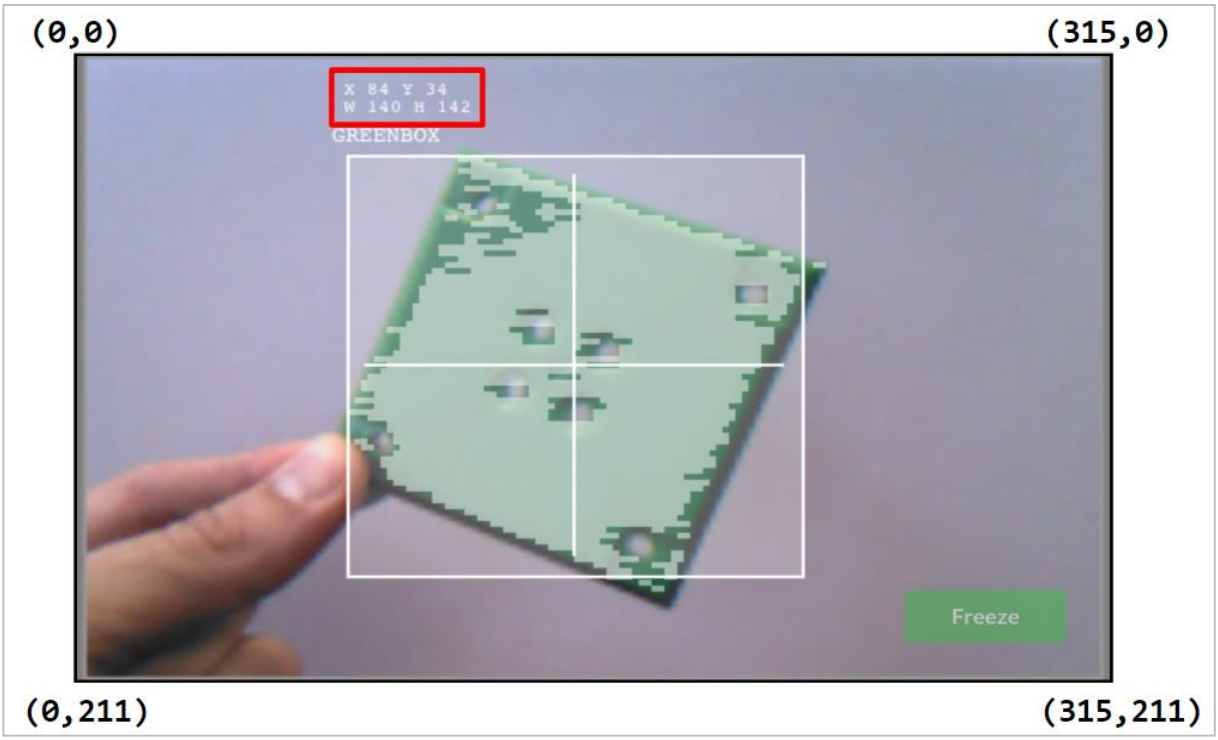

- The *object count* instruction tells us how many GREENBOX objects are in the snapshot. Here, there is only 1 detected.
- The center X value tells us whether the GREENBOX object is to the left or right of the robot's center point. Remember, the Vision Sensor should be mounted in the middle of the robot facing forward and so the snapshot's view is the robot's view.
  - If center X is greater than 157.5, the object is to the right of the robot's center point.
  - If center X is less than 157.5, the object is to the left of the robot's center point.
- The center Y value tells us whether the GREENBOX is higher or lower than the robot's center point.
  - If center Y is greater than 105.5, the object is lower than the robot's center point.
  - If center Y is less than 105.5, the object is higher than the robot's center point.
- The width and height values tell us how close the GREENBOX is to the robot.

- The same-sized object will be larger in width and height as it gets closer to the robot.

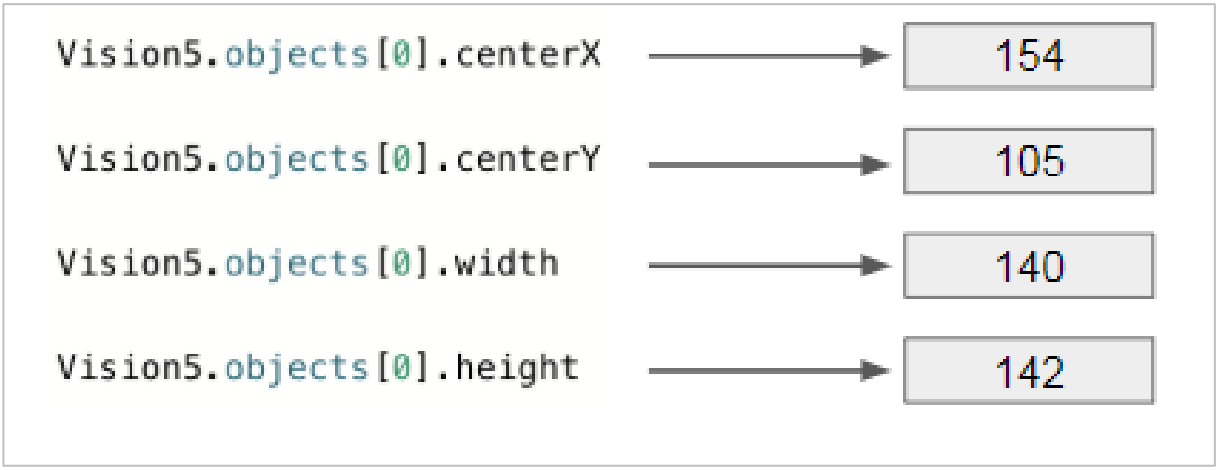
## How are the center X and center Y values calculated?

The values are calculated based on the coordinates within the snapshot. The width and height of the object are already calculated.

The Vision Sensor tracks the X and Y values of the upper left corner of the object. Below, those coordinates are (84, 34).



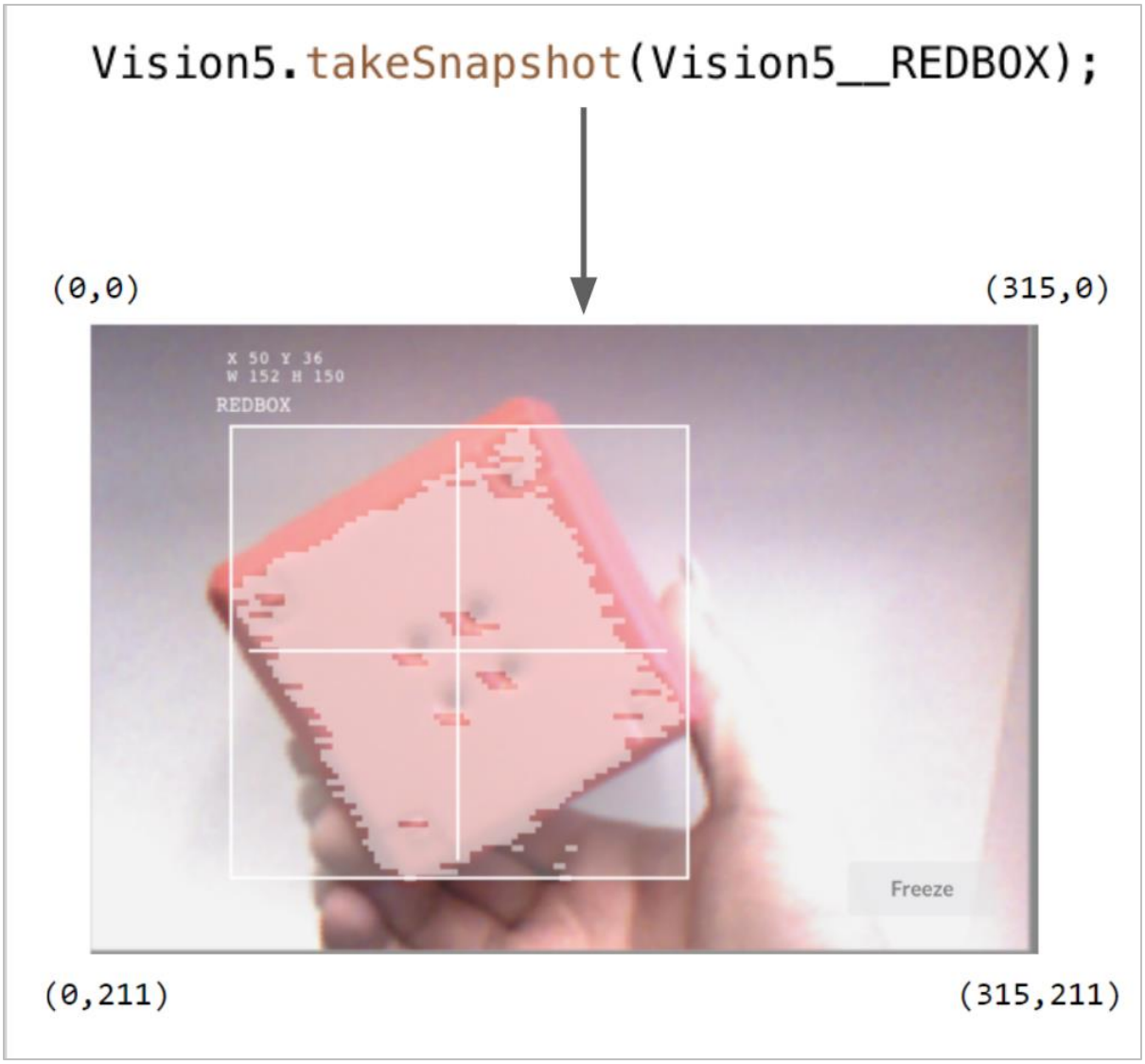

The center X and center Y values can be calculated based off of the coordinates of the upper left corner (84, 34), and the width (W 140) and height (H 142) values provided.




- centerX = 140/2 + 84 = **154**

    - centerX = half the width of the object added to its leftmost X coordinate
- centerY = 142/2 + 34 = **105**
    - centerY = half the height of the object added to its topmost Y coordinate

# Practice for the Vision Data Challenge - VEXcode V5 Text

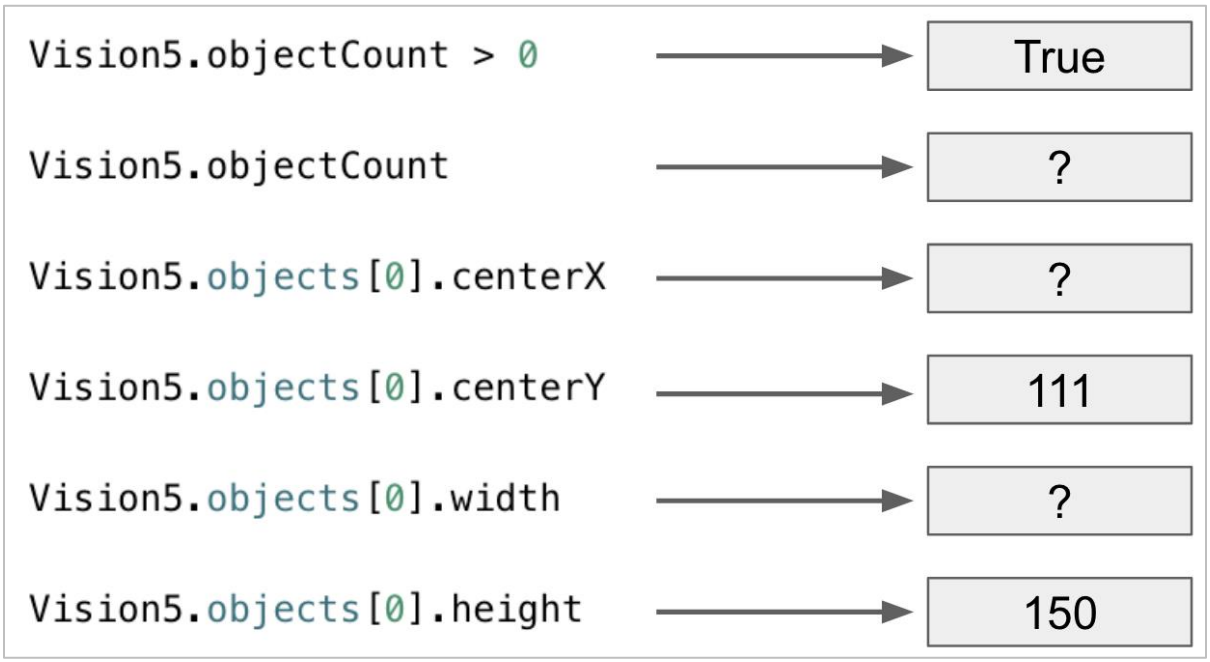



## Add the missing values below in your engineering notebook.

Here are the provided data from the snapshot:

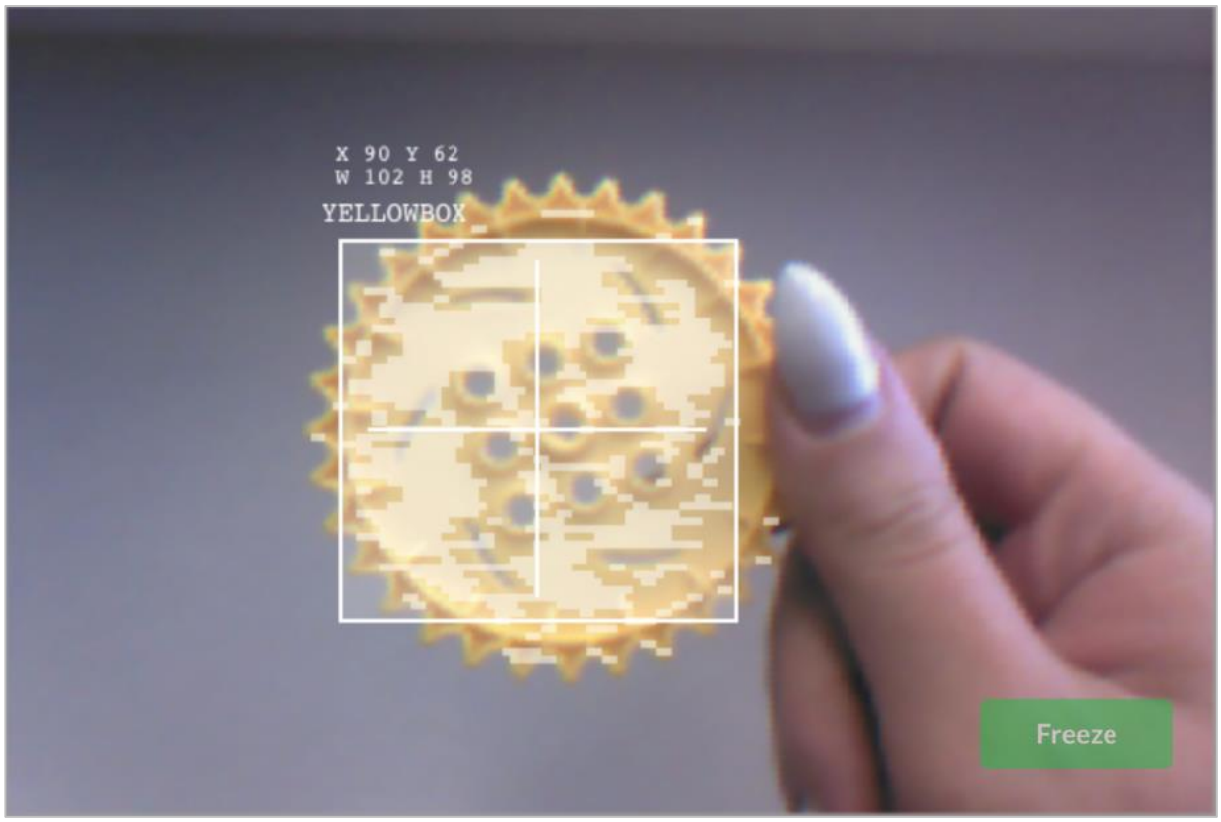
- X = 50
- Y = 36
- W = 152

- H = 150



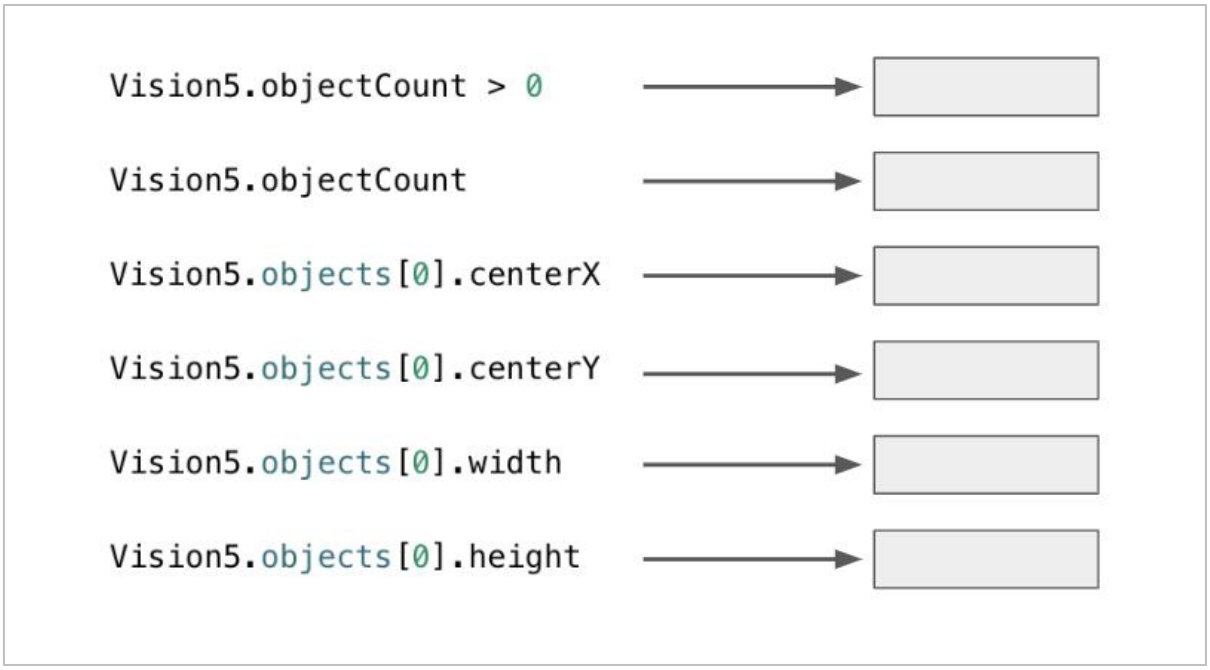

- Is the REDBOX to the left or to the right of the robot's center point?
- Is the REDBOX higher or lower than the robot's center point?

# The Vision Data Challenge - VEXcode V5 Text




## Complete the Vision Data Challenge by answering the questions and filling in the missing data in your engineering notebook.

- Which of these instructions was used to take the snapshot above?
  - 
  - 
- Fill in these values:

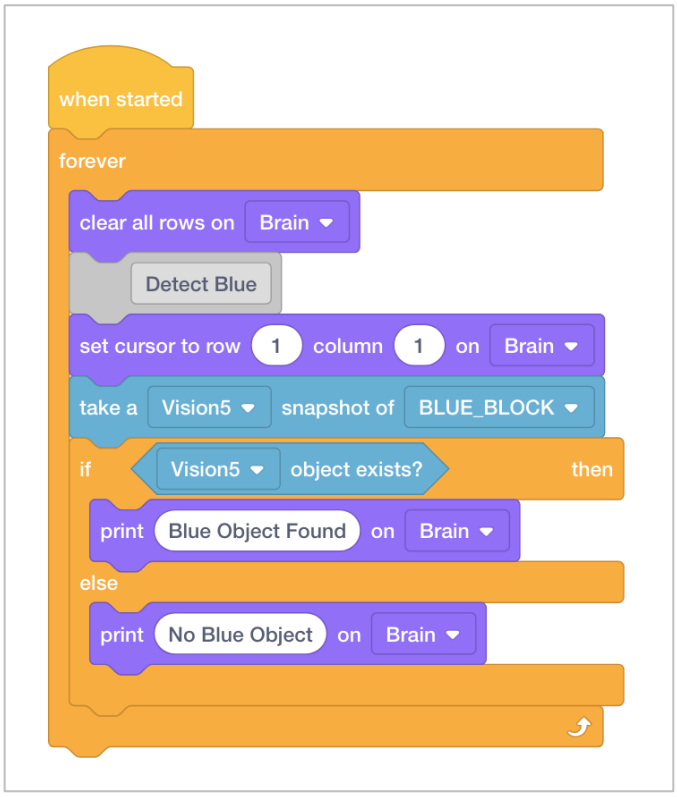



- Is YELLOWBOX to the left or to the right of the robot's center point?
- Is YELLOWBOX above or below the robot's center point?
- YELLOWBOX is **NOT** the best name to give this object if you want to easily recognize which color signature is which. Which of these is a better name? Why?
    - YELLOWGEAR
    - YELLOWCUBE


Understand the core concepts and how to apply them to different situations. This review process will fuel motivation to learn.

# Review - VEXcode V5 Blocks

1. **What does the `snapshot` block do in this example?**




   - Streams a continuous video of what the Vision Sensor sees
   - Prints the color of the object on the Brain's screen
   - Determines of an object exists or not
   - Takes a snapshot of the current image from the Vision Sensor so that it can be analyzed

2. **Which of the following should come first when configuring a signature for the Vision Sensor?**

   - Still the image so an area of color can be selected
   - Select the "Set" button
   - Clear the signature
   - Place the object in view of the Vision Sensor

3. **Which of the following is NOT an example of tuning the Vision Sensor?**
    - Adjusting the color signature slider
    - Adjusting the brightness
    - Resetting the color signature of adjusting the slider and brightness does not help
    - Calculating the center x

4. **Why is a `forever` block used in the Detecting Objects example project?**
    - "Blue object found" should be printed forever
    - The snapshot block only takes one current image of what the Vision Sensor sees. Using the forever block allows the Vision Sensor to take multiple snapshots so that it can continuously check for different objects.
    - The blocks inside should only repeat a certain number of times
    - The if-then-else block needed to be contained inside another looping block

5. **The leftmost X value of an object is 30 pixels and the width is 40 pixels in total. What is true about this object?**
    - Its centerX is 50 and it is to the left of the robot's center point.
    - Its centerX is 70 and it is to the left of the robot's center point.
    - Its centerX is 50 and it is to the right of the robot's center point.
    - Its centerX is 70 and it is to the right of the robot's center point.

# Review - VEXcode V5 Text

6. **What does the takeSnapshot() instruction do?**
   - Streams a continuous video of what the Vision Sensor sees
   - Prints the color of the object on the Brain's screen
   - Determines of an object exists or not
   - Takes a snapshot of the current image from the Vision Sensor so that it can be analyzed

7. **Which of the following should come first when configuring a signature for the Vision Sensor?**
   - Still the image so an area of color can be selected
   - Select the "Set" button
   - Clear the signature
   - Place the object in view of the Vision Sensor

8. **Which of the following is NOT an example of tuning the Vision Sensor?**
   - Adjusting the color signature slider
   - Adjusting the brightness
   - Resetting the color signature of adjusting the slider and brightness does not help
   - Calculating the center x

9. **Why is a `forever` block used in the Detecting Objects example project?**
   - "Blue object found" should be printed forever
   - The takeSnapshot() only takes one current image of what the Vision Sensor sees. Using the forever structure allows the Vision Sensor to take multiple snapshots so that it can continuously check for different objects.
   - The instructions inside should only repeat a certain number of times
   - The if-then-else structure needed to be contained inside another looping structure

10. **The leftmost X value of an object is 30 pixels and the width is 40 pixels in total. What is true about this object?**

    - Its centerX is 50 and it is to the left of the robot's center point.
    - Its centerX is 70 and it is to the left of the robot's center point.
    - Its centerX is 50 and it is to the right of the robot's center point.
    - Its centerX is 70 and it is to the right of the robot's center point.

# APPENDIX

Additional information, resources, and materials.

# Using the 1 Post Hex Nut Retainer w/ Bearing Flat

*1 Post Hex Nut Retainer w/ Bearing Flat*

## Using the 1 Post Hex Nut Retainer w/ Bearing Flat

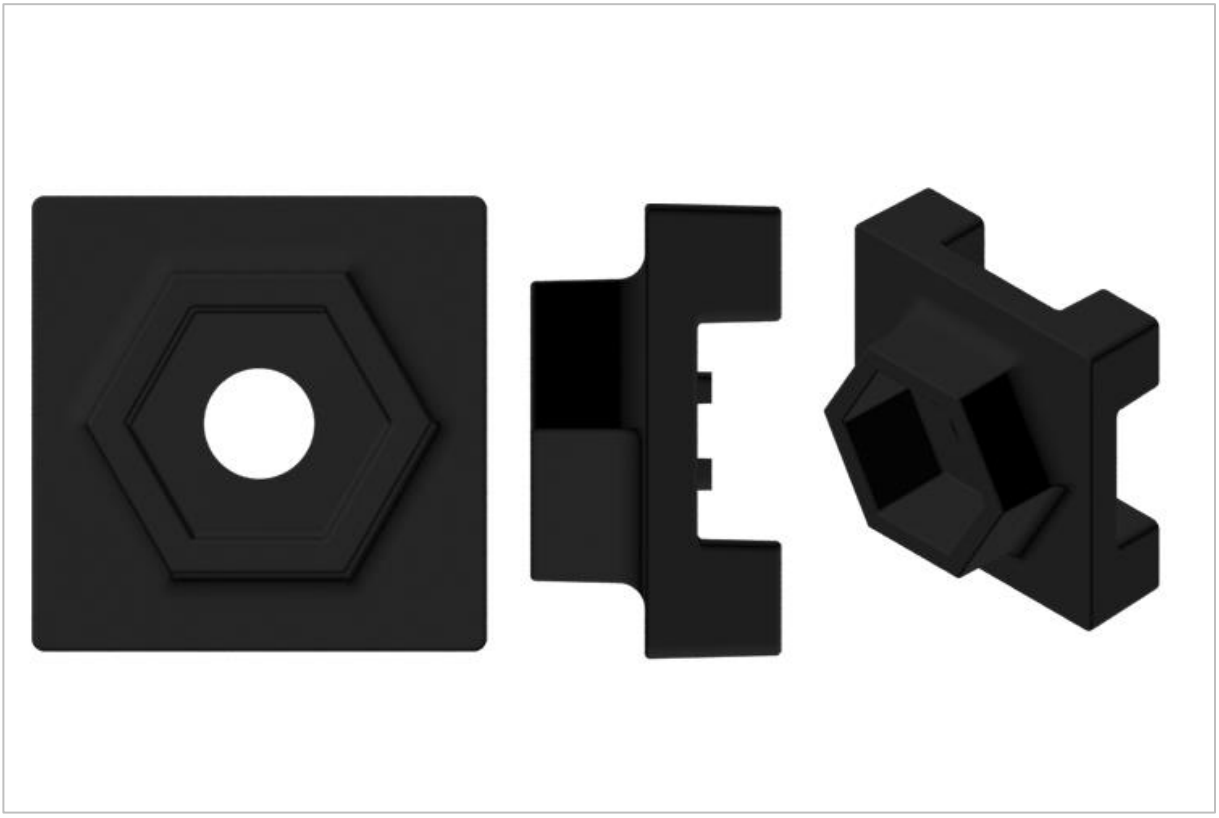
The 1 Post Hex Nut Retainer w/ Bearing Flat allows shafts to spin smoothly through holes in structural components. When mounted, it provides two points of contact on structural components for stability. One end of the retainer contains a post sized to securely fit in the square hole of a structural component. The center hole of the retainer is sized and slotted to securely fit a hex nut, allowing a 8-32 screw to easily be tightened without the need for a wrench or pliers. The hole on the end of the Retainer is intended for shafts or screws to pass through.

To make use of the retainer:

- Align it on a VEX structural component such that the end hole is in the desired location, and the center and end sections are also backed by the structural component.

- Insert the square post extruding from the retainer into the structural component to help keep it in place.
- Insert a hex nut into the center section of the retainer so that it is flush with the rest of the component.
- Align any additional structural components to the back of the main structural component, if applicable.
- Use an 8-32 screw of appropriate length to secure the structural component(s) to the retainer through the center hole and hex nut.

# Using the 4 Post Hex Nut Retainer



*4 Post Hex Nut Retainer*

## Using the 4 Post Hex Nut Retainer

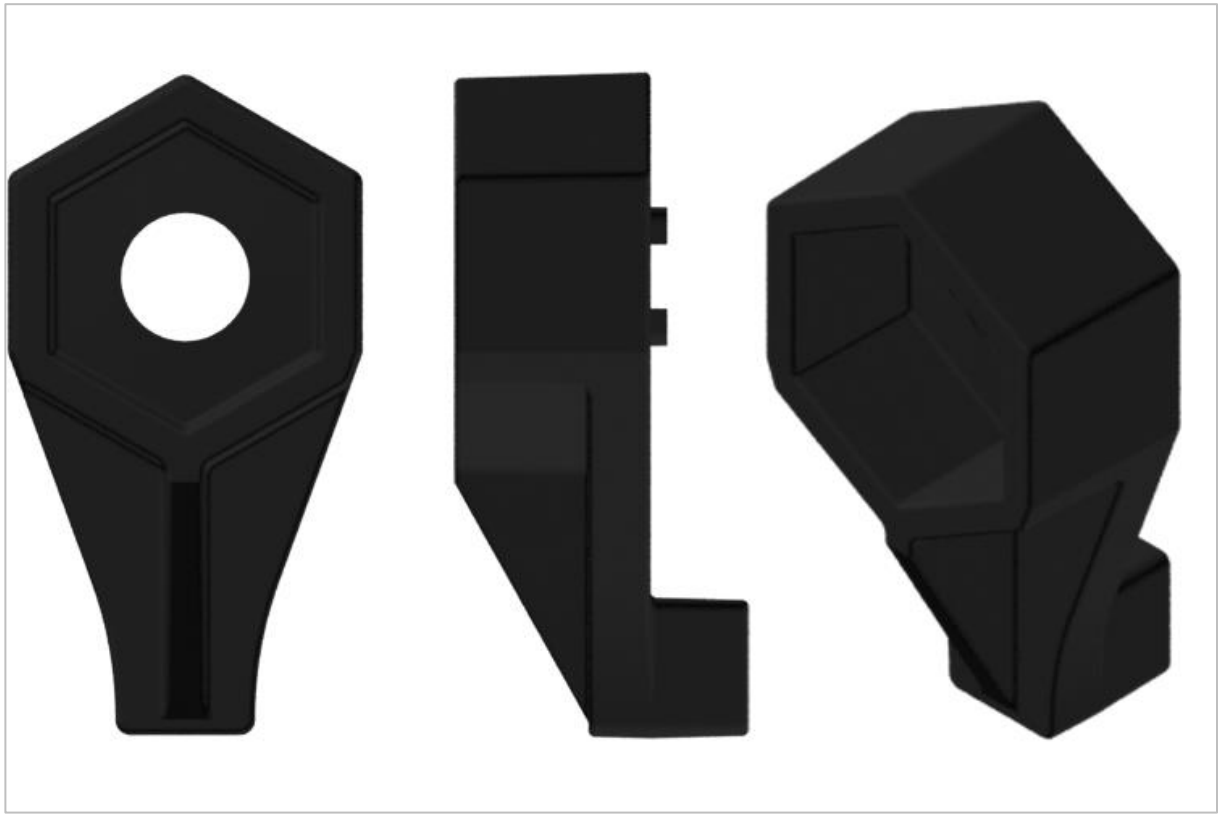
The 4 Post Hex Nut Retainer provides five points of contact for creating a strong connection between two structural components using one screw and nut. Each corner of the retainer contains a post sized to securely fit in a square hole within a structural component. The center of the retainer is sized and slotted to securely fit a hex nut, allowing a 8-32 screw to easily be tightened without the need for a wrench or pliers.

To make use of the retainer:

- Align it on a VEX structural component such that the center hole is in the desired location, and each corner is also backed by the structural component.
- Insert the square posts extruding from the retainer into the structural component to help keep it in place.
- Insert a hex nut into the center section of the retainer so that it is flush with the rest of the component.

- Align any additional structural components to the back of the main structural component, if applicable.
- Use an 8-32 screw of appropriate length to secure the structural component(s) to the retainer through the center hole and hex nut.

# Using the 1 Post Hex Nut Retainer



*1 Post Hex Nut Retainer*

## Using the 1 Post Hex Nut Retainer

The 1 Post Hex Nut Retainer provides two points of contact for connecting a structural component to another piece using one screw and nut. One end of the retainer contains a post sized to securely fit in the square hole of a structural component. The other end of the retainer is sized and slotted to securely fit a hex nut, allowing a 8-32 screw to easily be tightened without the need for a wrench or pliers.

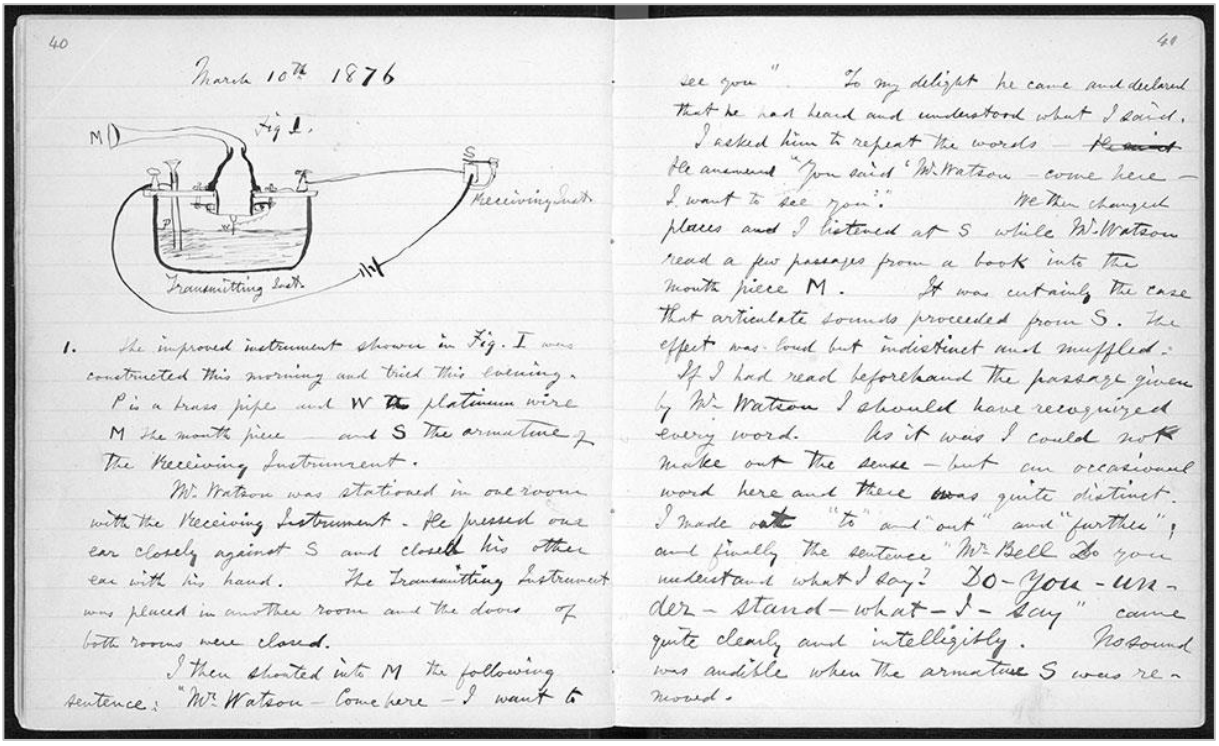
To make use of the retainer:

- Align it on a VEX structural component such that both ends are backed by the structural component and positioned to secure the second piece.
- Insert the square post extruding from the retainer into the structural component to help keep it in place.
- If the retainer is being used to secure two structural components, insert a hex nut into the other end of the retainer so that it is flush with the rest of the component. If used to secure

a different type of component, such as a standoff, it may be appropriate to insert the screw through this side.

- Align any additional components to the back of the main structural component, if applicable.
- If the retainer is being used to connect two structural components, use an 8-32 screw of appropriate length to secure the structural components through the hole and hex nut. If used to connect a different type of component, such as a standoff, secure it directly or with a hex nut.

# Engineering Notebooks



40

March 10th 1876

Fig 1.

M

S

Receiving Instr.

Transmitting Instr.

1. The improved instrument shown in Fig. I was constructed this morning and tried this evening. P is a brass pipe and W the platinum wire M the mouth piece — and S the armature of the Receiving Instrument.

Mr. Watson was stationed in one room with the Receiving Instrument - He pressed one ear closely against S and closed his other ear with his hand. The Transmitting Instrument was placed in another room and the doors of both rooms were closed.

I then shouted into M the following sentence: "Mr. Watson — Come here — I want to

41

see you" To my delight he came and declared that he had heard and understood what I said.

I asked him to repeat the words He answered "You said 'Mr. Watson — come here — I want to see you'."

We then changed places and I listened at S while Mr. Watson read a few passages from a book into the mouth piece M. It was certainly the case that articulate sounds proceeded from S. The effect was loud but indistinct and muffled.

If I had read beforehand the passage given by Mr. Watson I should have recognized every word. As it was I could not make out the sense — but an occasional word here and there was quite distinct. I made out "to" and "out" and "further", and finally the sentence "Mr. Bell Do you understand what I say? DO-YOU-UN-DER-STAND-WHAT-I-SAY" came quite clearly and intelligibly. No sound was audible when the armature S was removed.

*Alexander Graham Bell's notebook entry from a successful experiment with his first telephone*

## An Engineering Notebook Documents your Work

Not only do you use an engineering notebook to organize and document your work, it is also a place to reflect on activities and projects. When working in a team, each team member will maintain their own journal to help with collaboration.

Your engineering notebook should have the following:

- An entry for each day or session that you worked on the solution
- Entries that are chronological, with each entry dated
- Clear, neat, and concise writing and organization
- Labels so that a reader understands all of your notes and how they fit into your iterative design process

An entry might include:

- Brainstorming ideas
- Sketches or pictures of prototypes

- Pseudocode and flowcharts for planning
- Any worked calculations or algorithms used
- Answers to guiding questions
- Notes about observations and/or conducted tests
- Notes about and reflections on your different iterations