To Do, or Not to Do

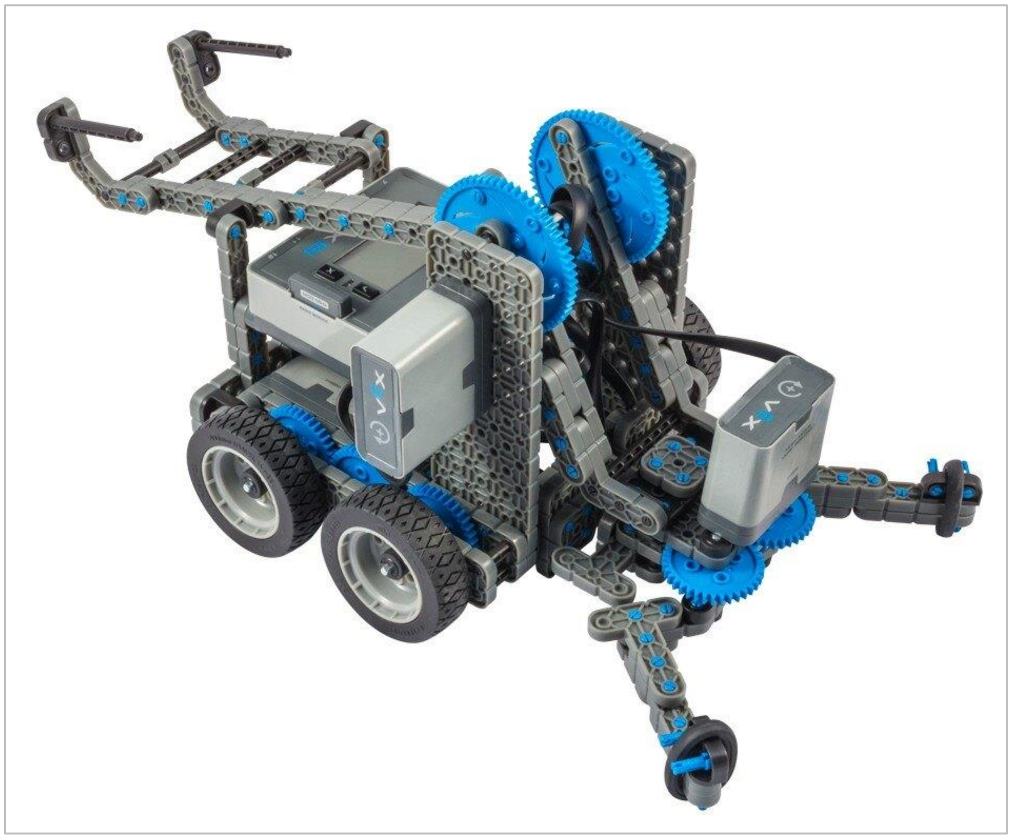
Explore the logic of if then else!

SPARK

SEEK

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

The Completed Look of the Build


The completed Clawbot IQ build.

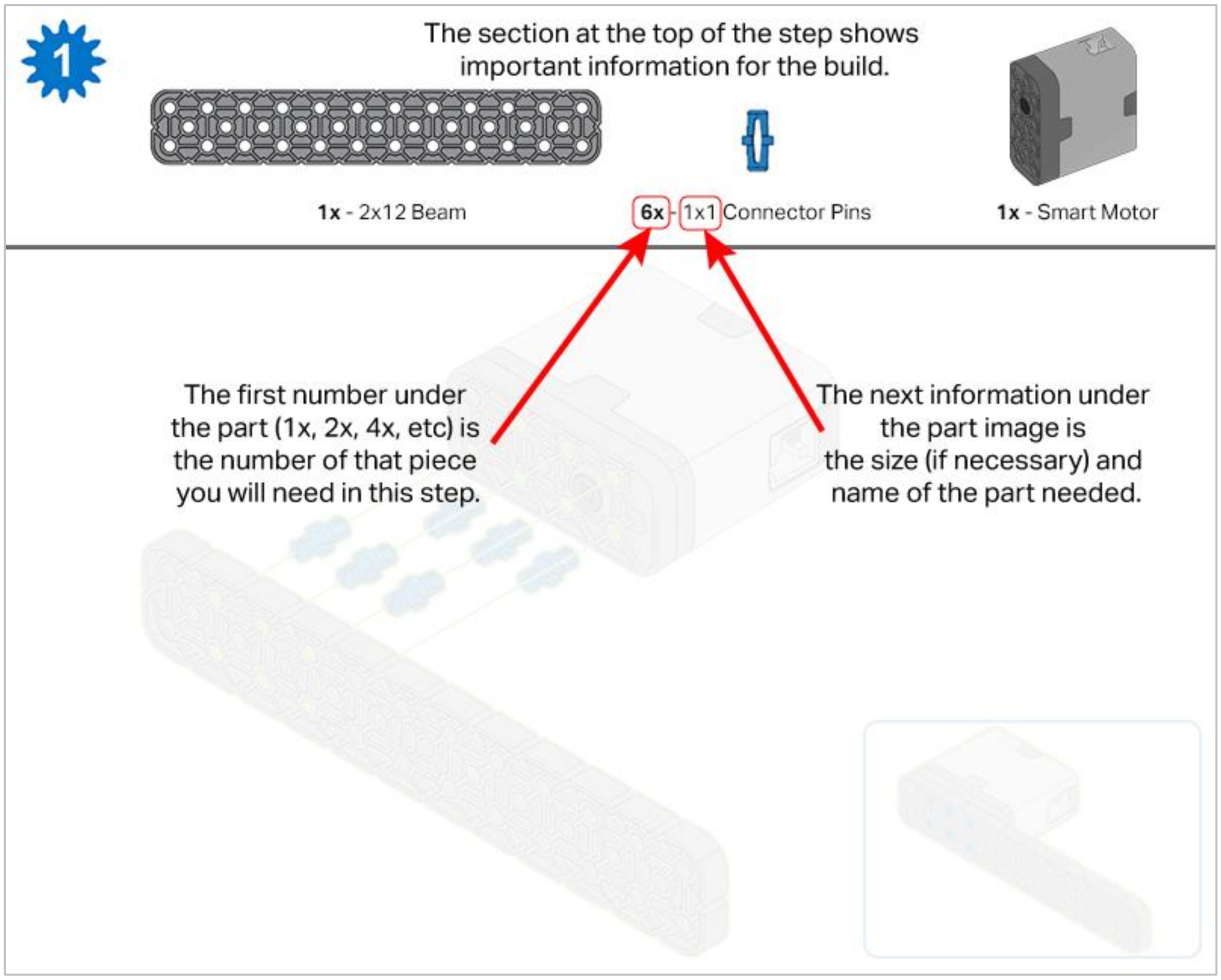
This robot is designed so that it can be built quickly and drive around either autonomously or with the Controller in a short amount of time.

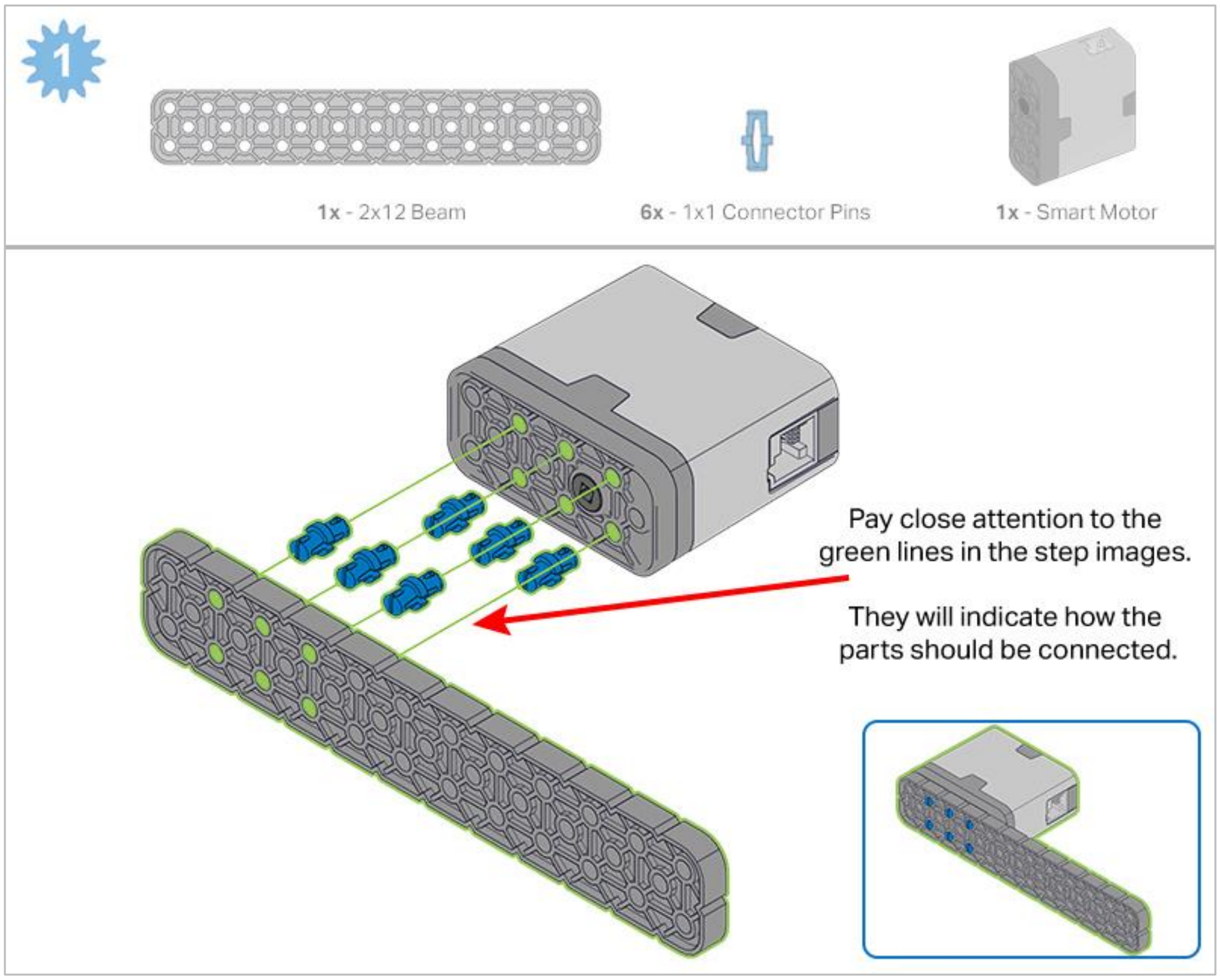
Build Instructions - Drivetrain + Distance

Build Instructions Summary

- **Drivetrain + Distance Building Instructions (19 steps):**
  - Right Wheel: steps 1 to 6
  - Left Wheel: steps 7 to 12
  - Distance Sensor: steps 13 to 19
- **Building Tips for All Steps:**
  - The section at the top of the step shows important information for the build. The first number under the image of the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step. The next information under the part image is the size and description of the part needed.
  - The finished step is illustrated in the box in the lower right corner.

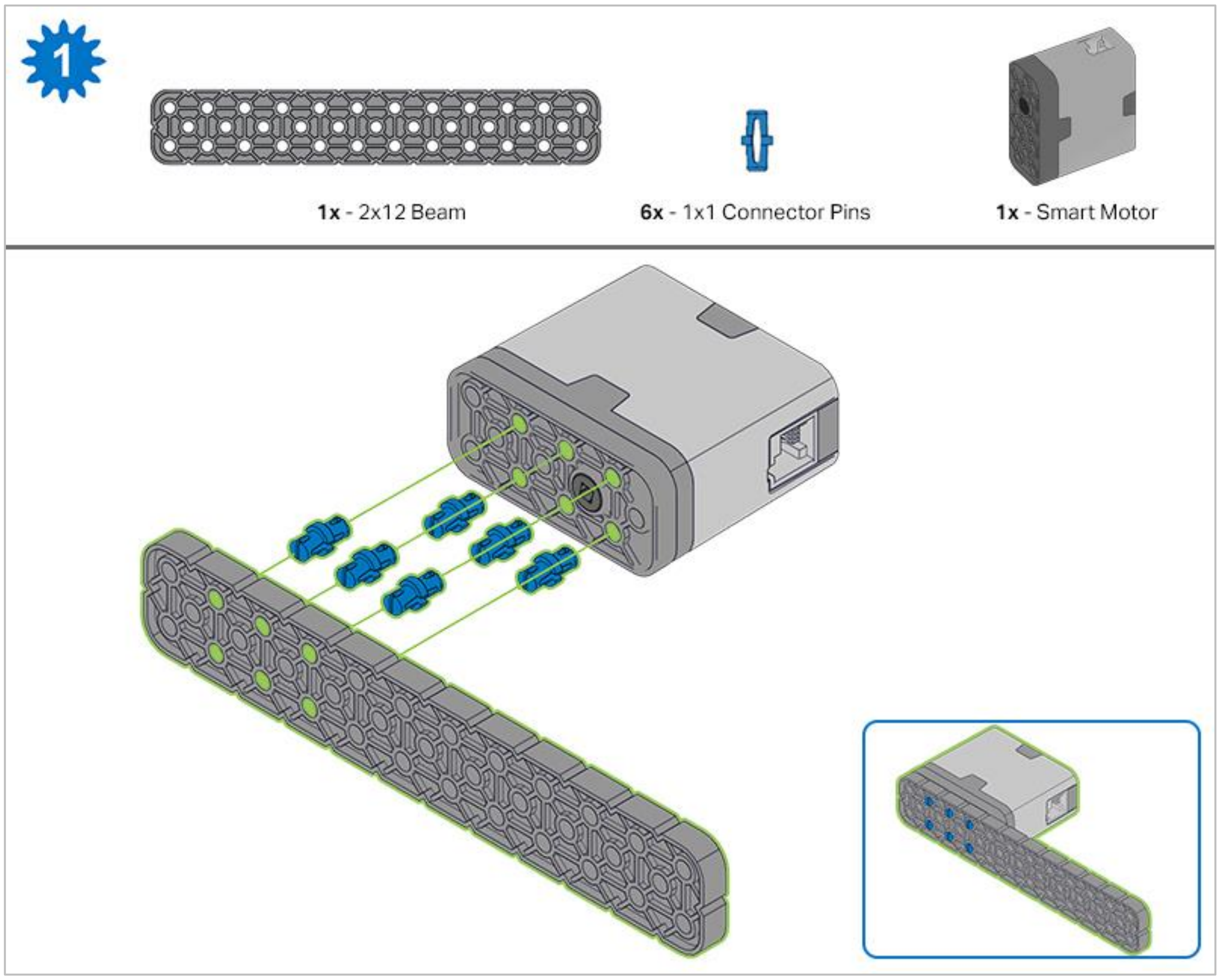
- Play close attention to the green lines in the step images. They will indicate how the parts should be connected.


1
The section at the top of the step shows important information for the build.
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
The first number under the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step.
The next information under the part image is the size (if necessary) and name of the part needed.

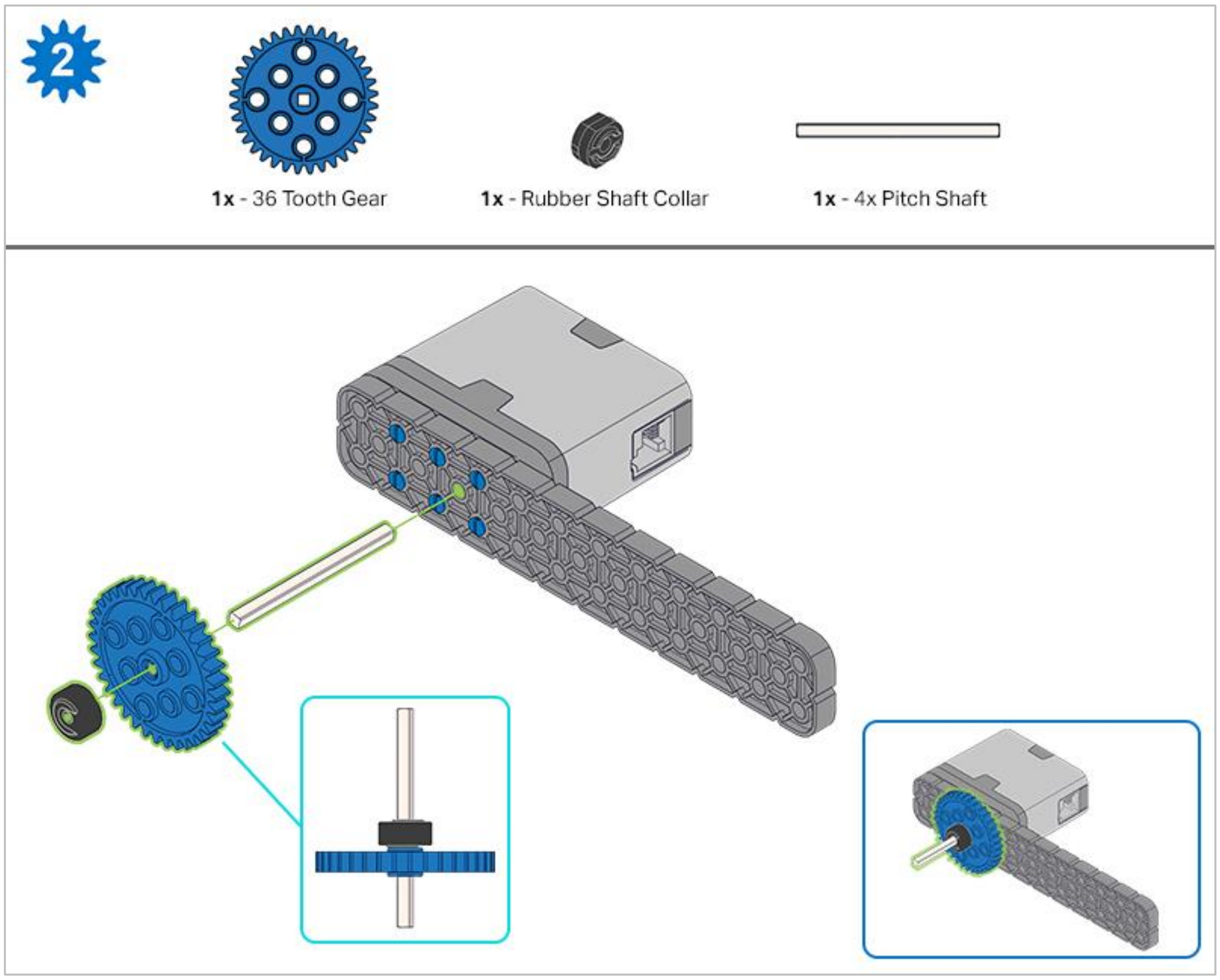


Pay close attention to the green lines in the step images.

They will indicate how the parts should be connected.



Step 1: Count all pieces before starting your build and have them readily available. Each team member should find the pieces for their section.



Step 2: When adding the 4x Pitch Shaft, twist the pitch shaft to check for tension while turning. If it spins freely, it is not properly inserted into the motor.

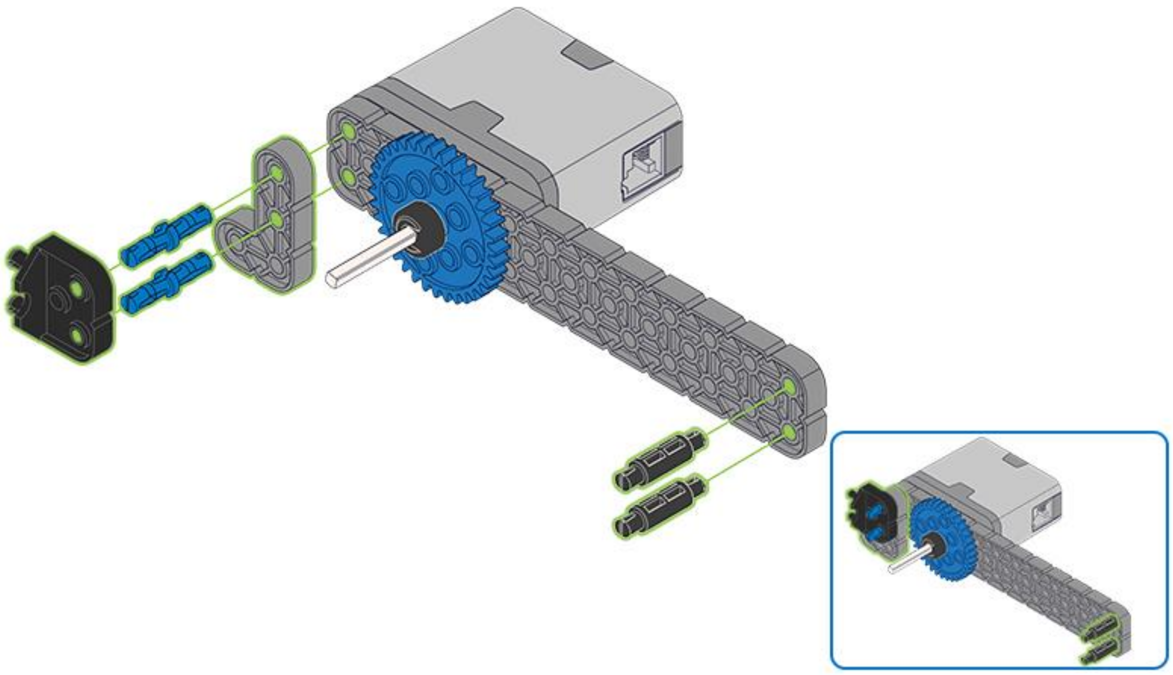
3

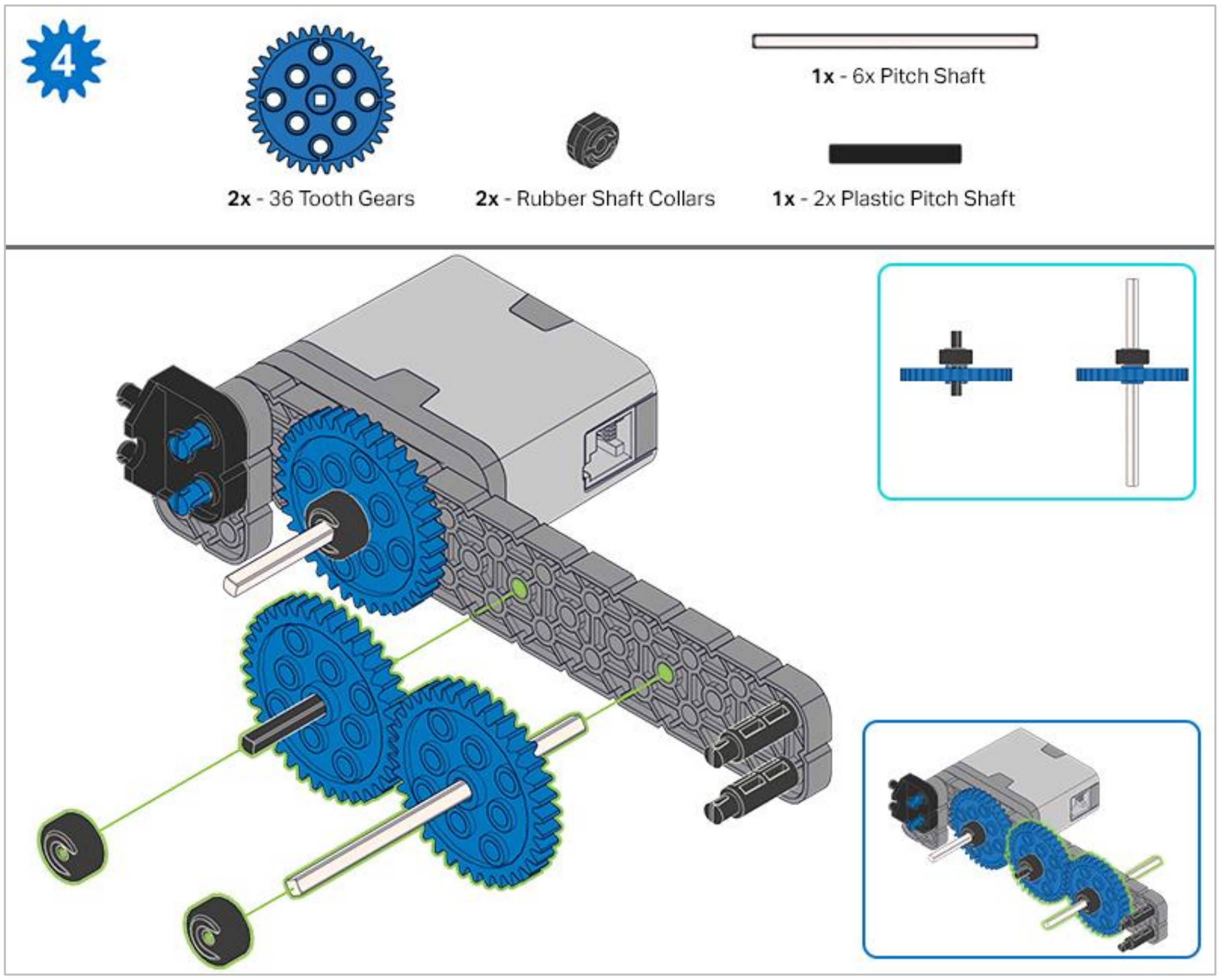
1x - 2x Wide, 1x2 Offset Corner Connector

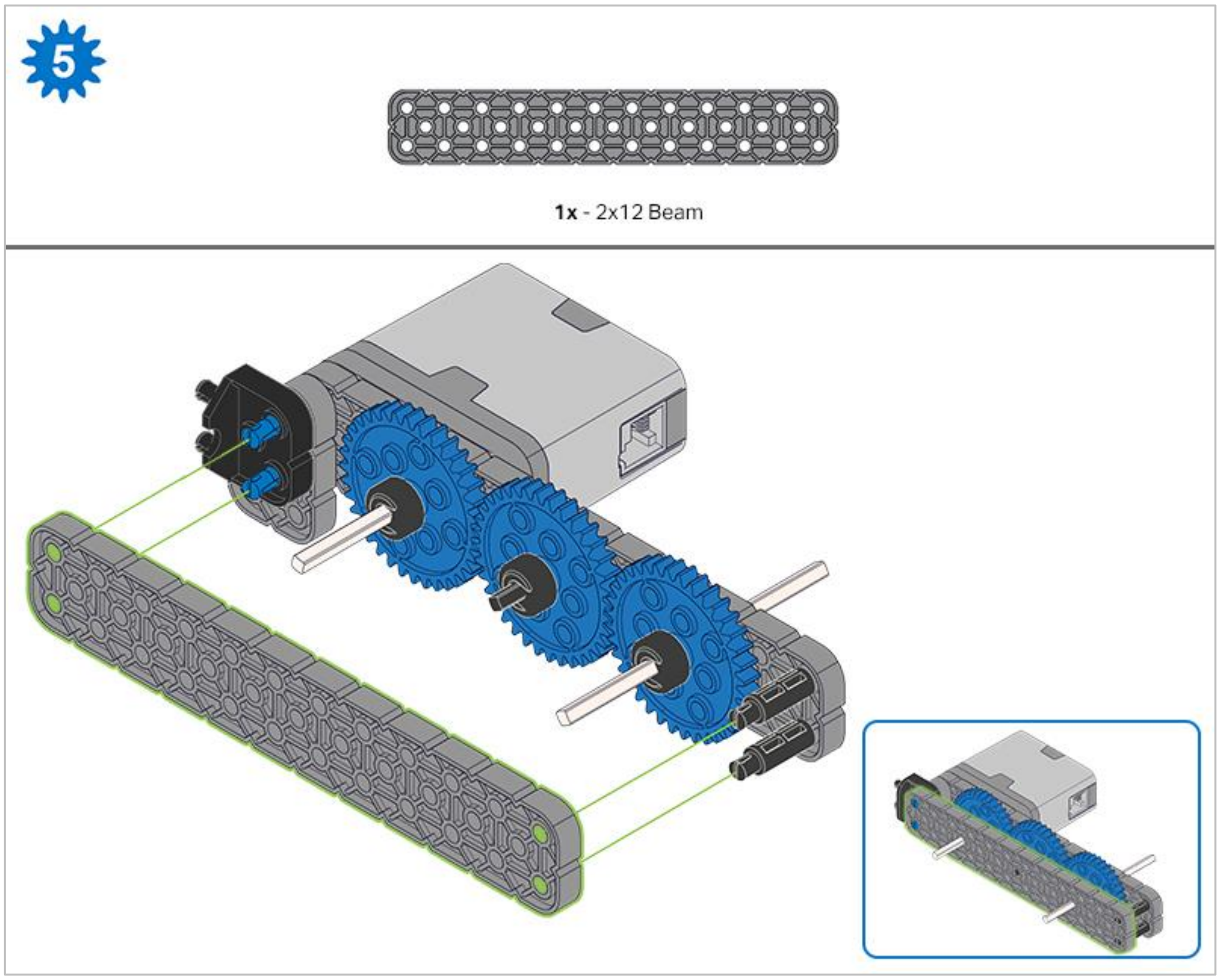
2x - 2x2 Connector Pins

1x - 2x3 Right Angle Beam

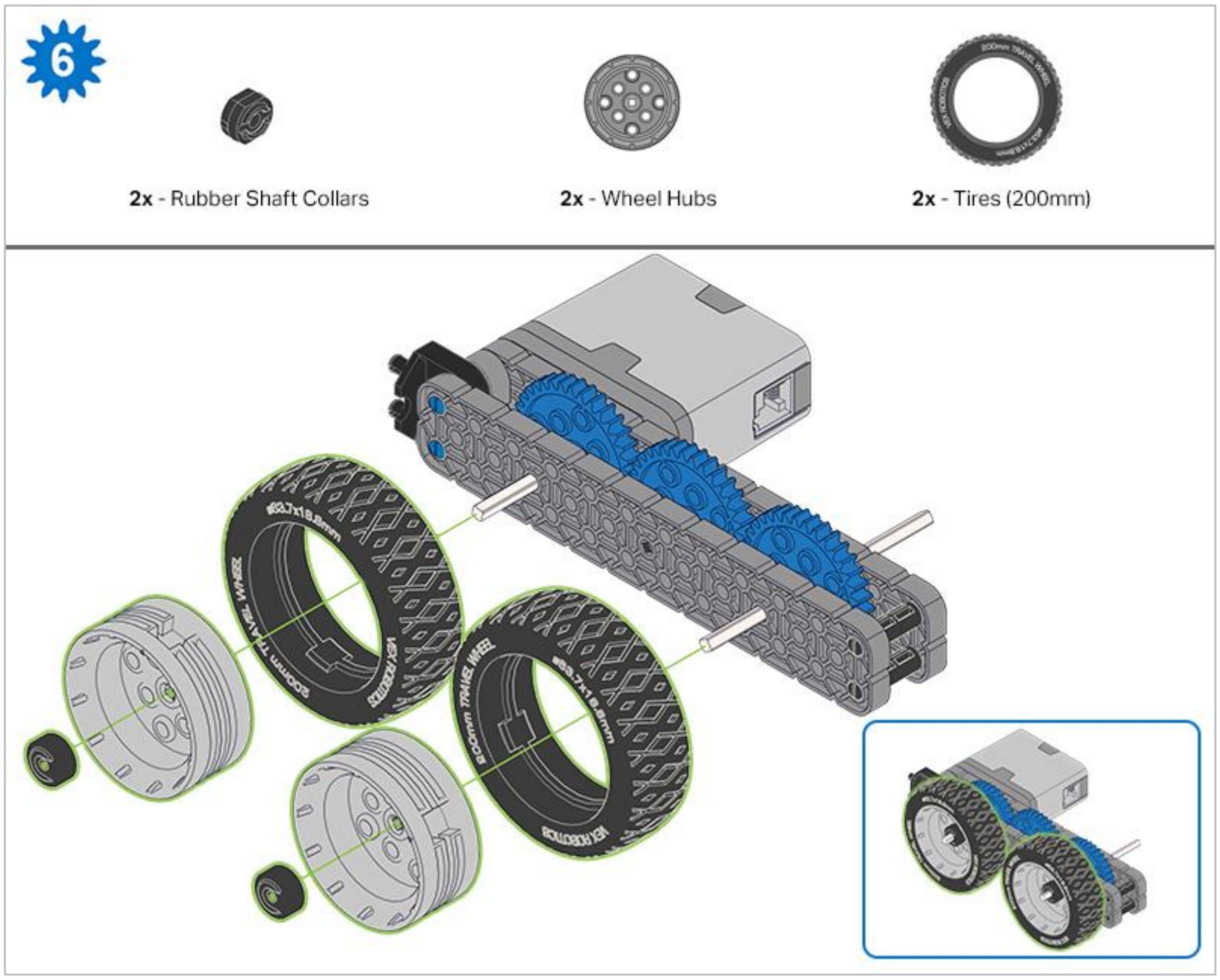
2x - 1x Pitch Standoffs



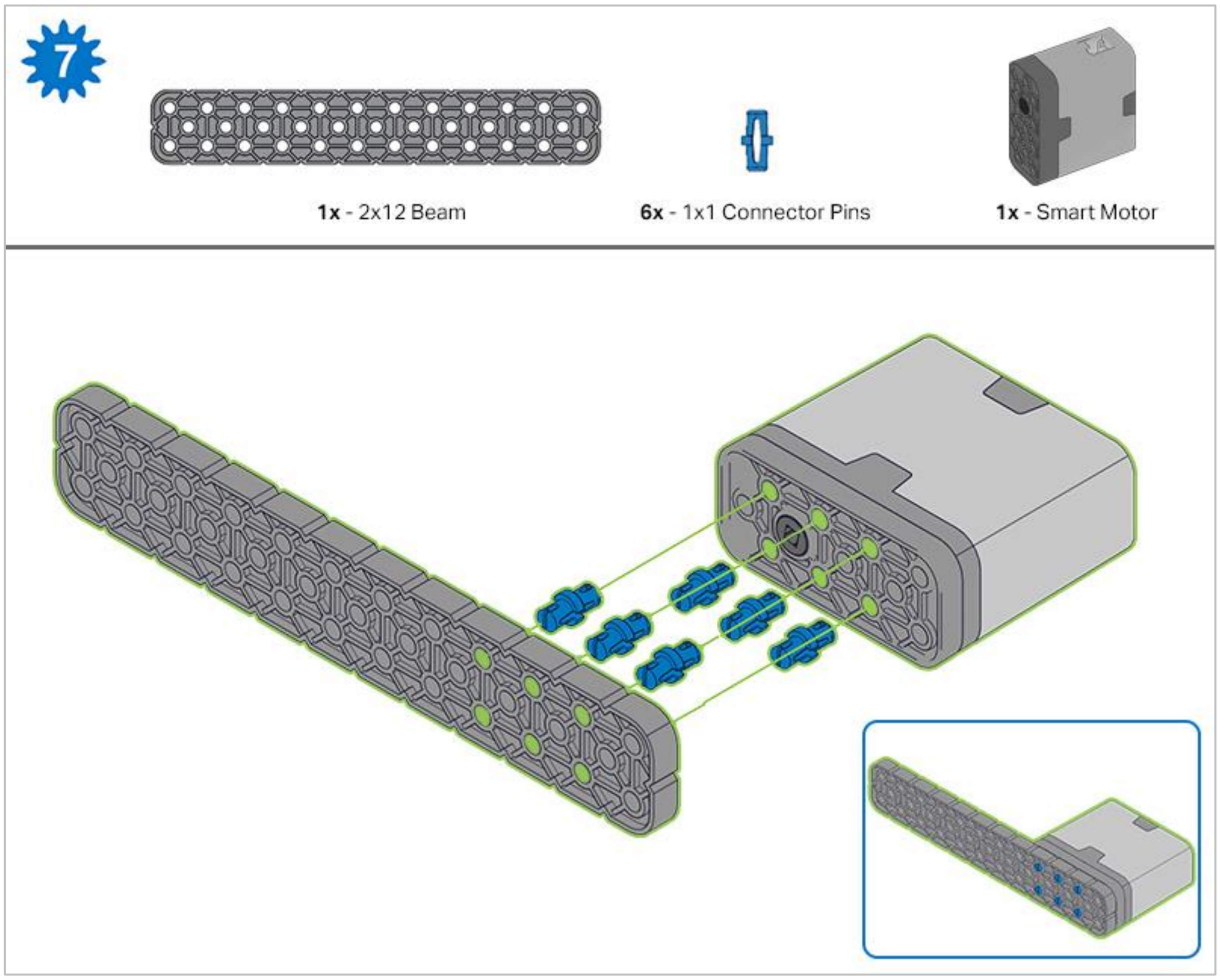

4
2x - 36 Tooth Gears
2x - Rubber Shaft Collars
1x - 6x Pitch Shaft
1x - 2x Plastic Pitch Shaft

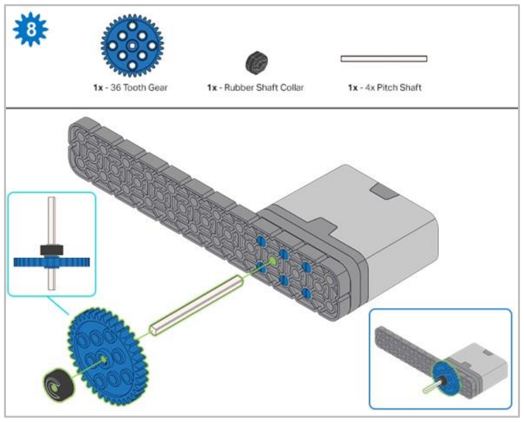


Step 5: Make sure the gears fit together properly before locking the 2x12 Beam in place.

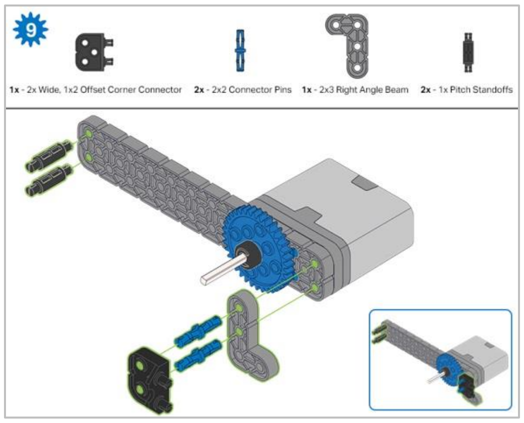


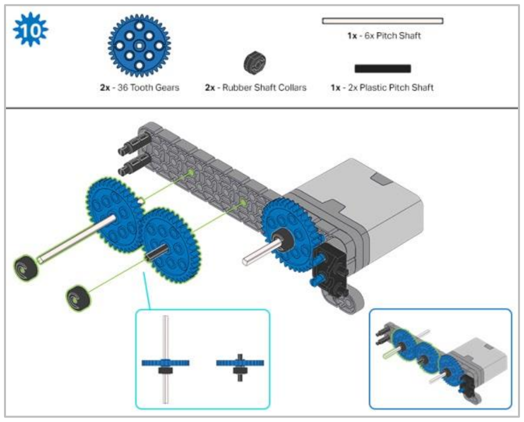
Step 6: After attaching the wheels, twist the wheel that has the shaft going into the motor. If the wheel spins freely and without tension, the 4x Pitch Shaft has slipped out of place.

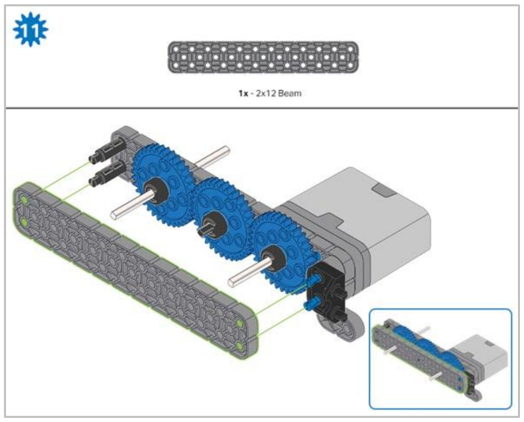




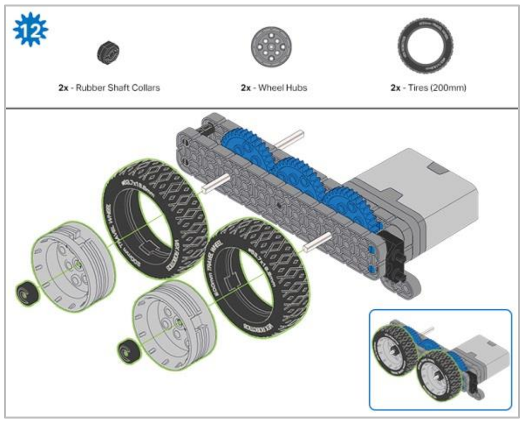
Step 8: When adding the 4x Pitch Shaft, twist the pitch shaft to check for tension while turning. If it spins freely, it is not properly inserted into the motor.


9
1x - 2x Wide, 1x2 Offset Corner Connector
2x - 2x2 Connector Pins
1x - 2x3 Right Angle Beam
2x - 1x Pitch Standoffs

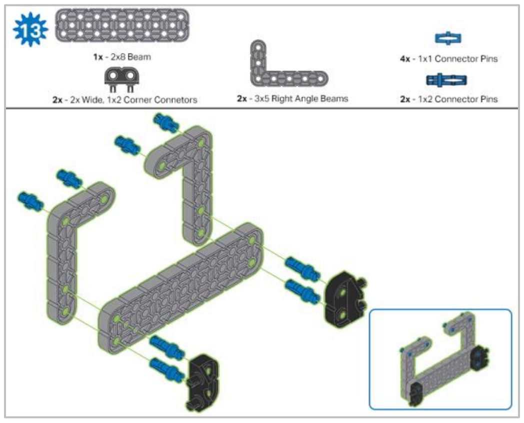


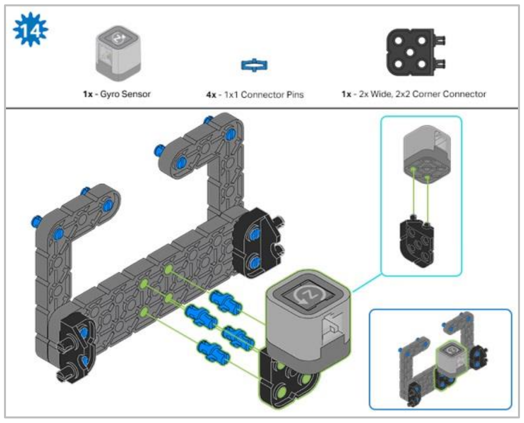


Step 11: Make sure the gears fit together properly before locking the 2x12 Beam in place.

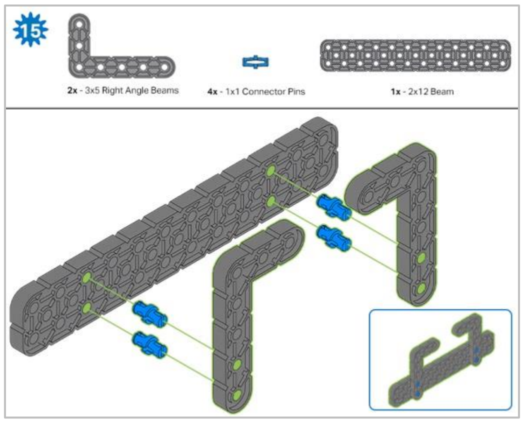


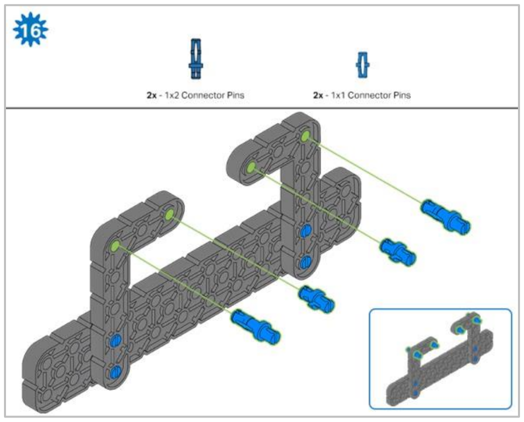
Step 12: After attaching the wheels, twist the wheel that has the shaft going into the motor. If the wheel spins freely and without tension, the 4x Pitch Shaft has slipped out of place.

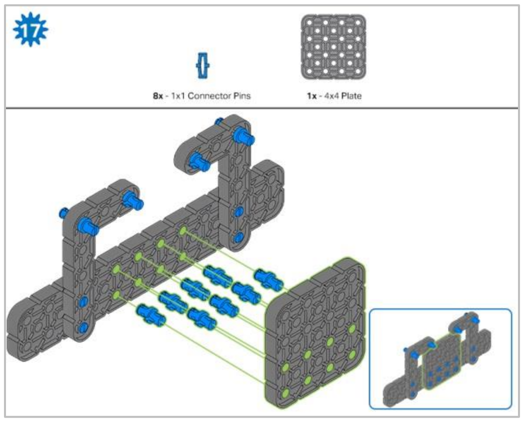


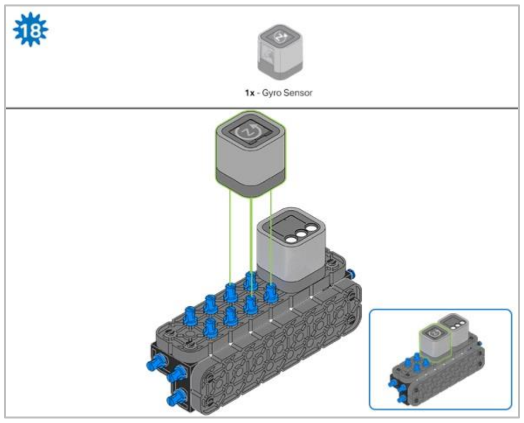


Step 14: Make sure the Gyro is placed the correct way to allow correct cable access.

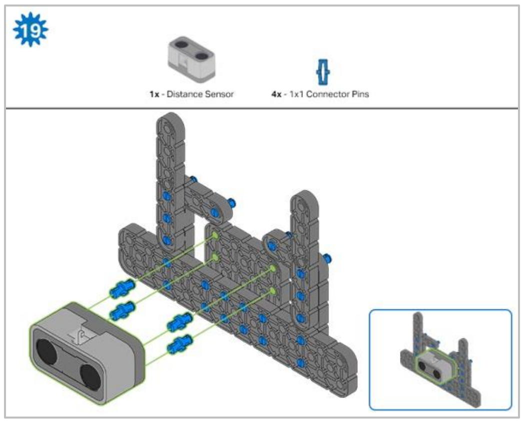








Step 18: The orange arrows mean spin the build around.



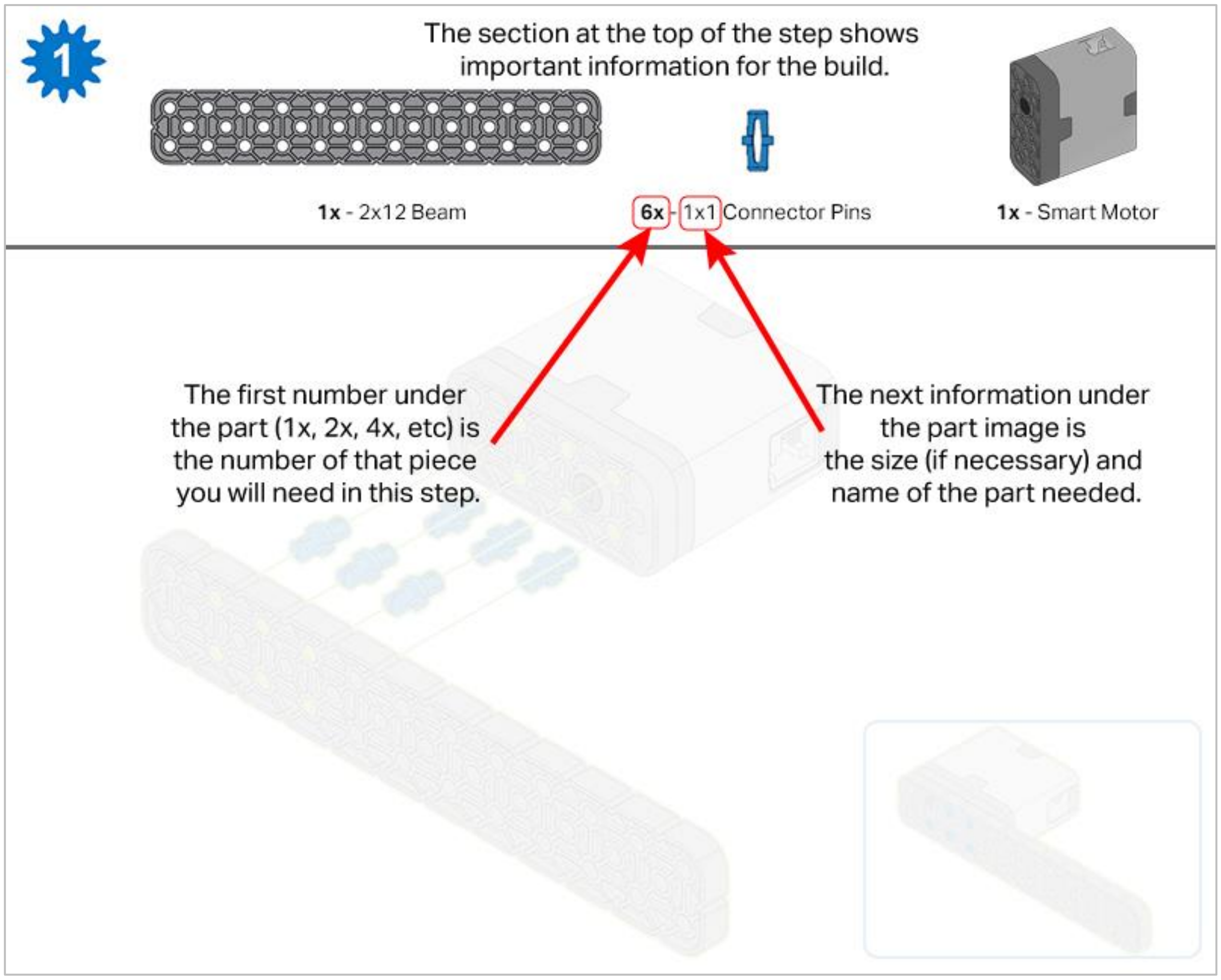
Step 19: When attaching the Distance Sensor, do not push on either of the two mesh covered openings. This will damage the sensor. Ensure the sensor is placed in the correct way to allow cable access.

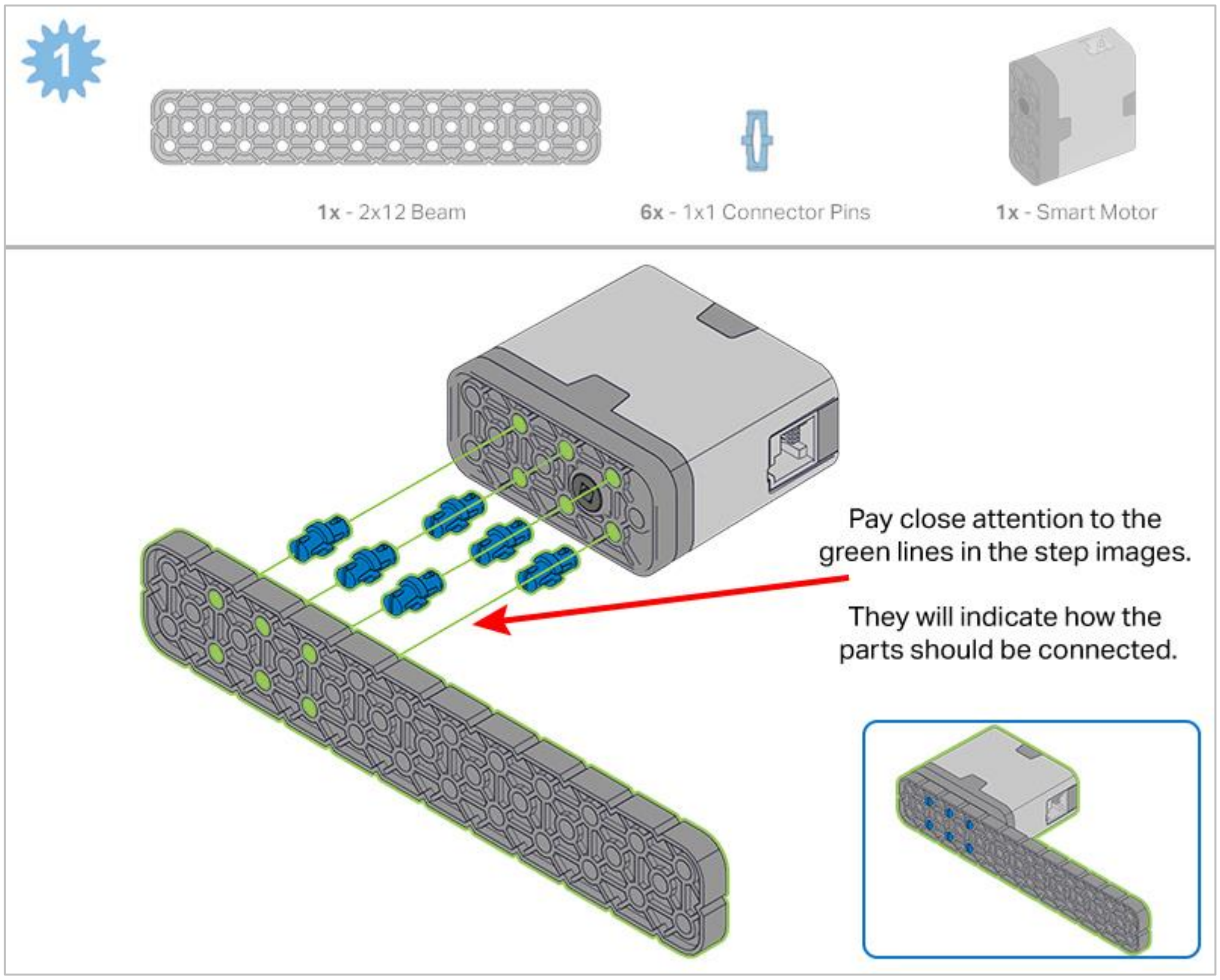
Build Instructions - Robot Frame

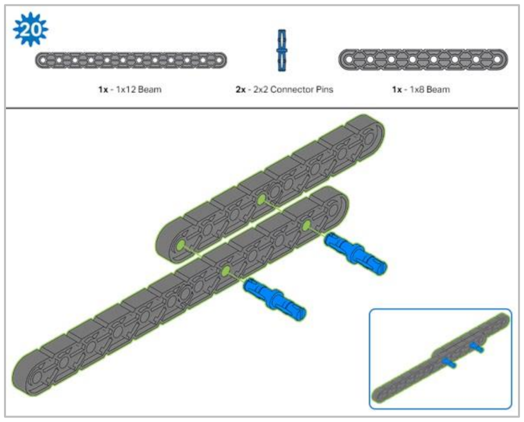
Build Instructions Summary

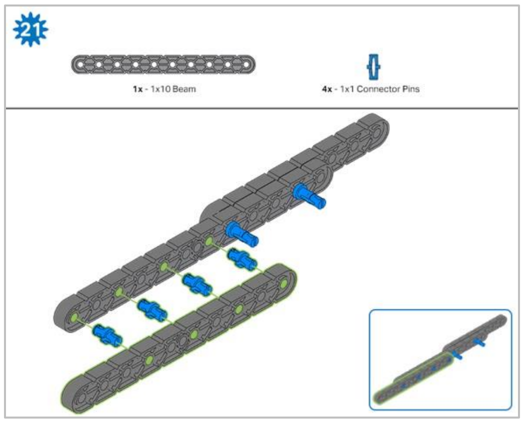
- **Robot Frame Building Instructions (22 steps):**
  - Cargo Holder: steps 20 to 28
  - Arm Base: steps 29 to 41
- **Building Tips for All Steps:**
  - The section at the top of the step shows important information for the build. The first number under the image of the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step. The next information under the part image is the size and description of the part needed.
  - The finished step is illustrated in the box in the lower right corner.

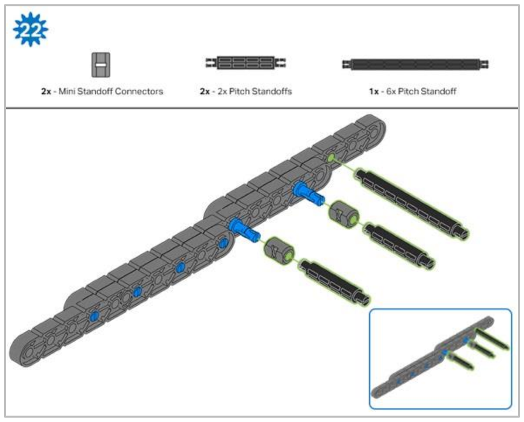
- Play close attention to the green lines in the step images. They will indicate how the parts should be connected.

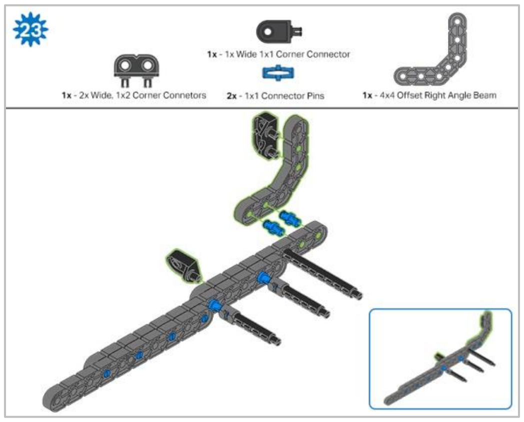

1
The section at the top of the step shows important information for the build.
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
The first number under the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step.
The next information under the part image is the size (if necessary) and name of the part needed.

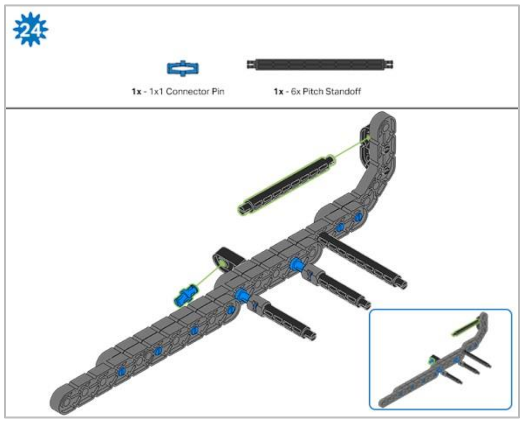

1
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
Pay close attention to the green lines in the step images.
They will indicate how the parts should be connected.


20
1x - 1x12 Beam
2x - 2x2 Connector Pins
1x - 1x8 Beam


21
1x - 1x10 Beam
4x - 1x1 Connector Pins


22
2x - Mini Standoff Connectors
2x - 2x Pitch Standoffs
1x - 6x Pitch Standoff

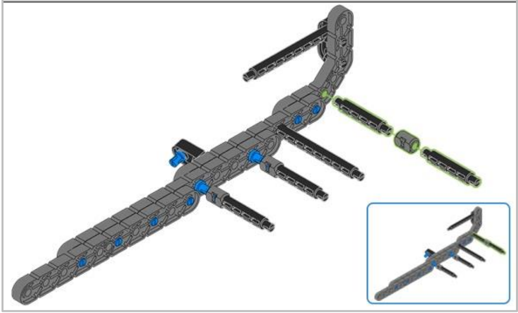

23
1x - 1x Wide 1x1 Corner Connector
1x - 2x Wide, 1x2 Corner Connetors
2x - 1x1 Connector Pins
1x - 4x4 Offset Right Angle Beam


24
1x - 1x1 Connector Pin
1x - 6x Pitch Standoff

25

1x - Mini Standoff Connector

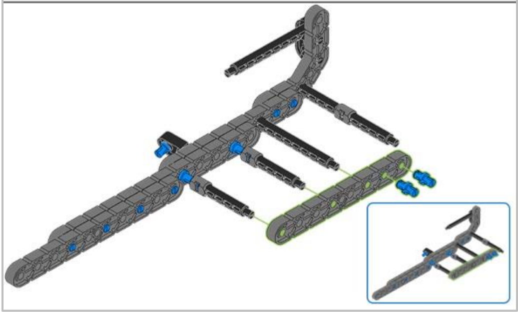
2x - 2x Pitch Standoffs



26

2x - 1x1 Connector Pins

1x - 1x8 Beam

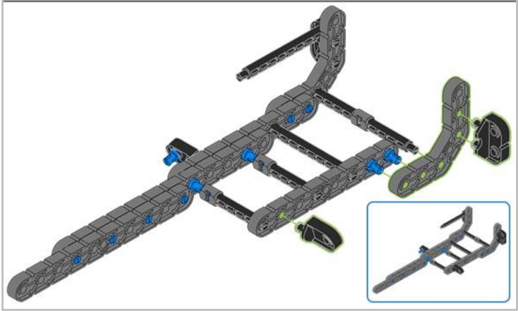


27

1x - 2x Wide, 1x2 Corner Connetors

1x - 1x Wide 1x1 Corner Connector

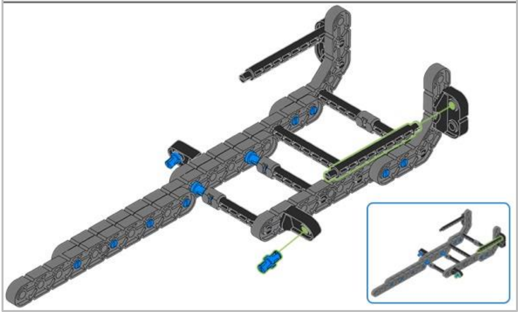
1x - 4x4 Offset Right Angle Beam

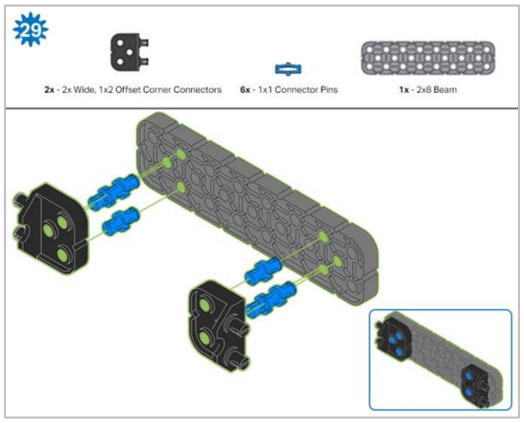


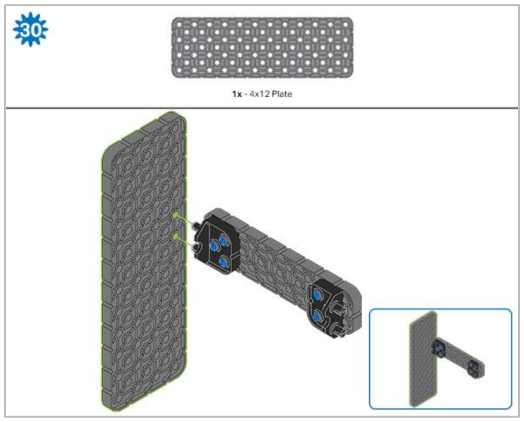
28

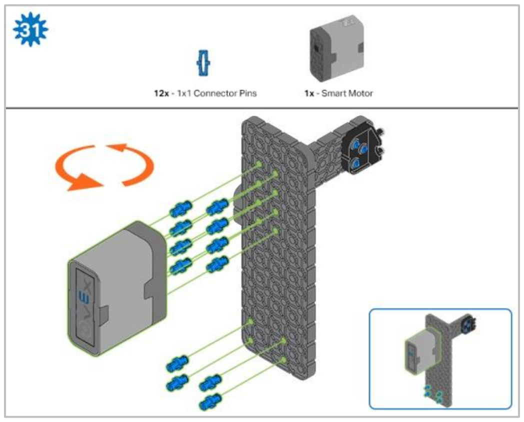
1x - 1x1 Connector Pin

1x - 6x Pitch Standoff

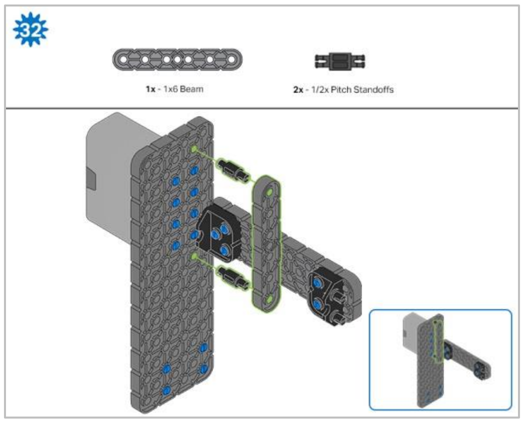


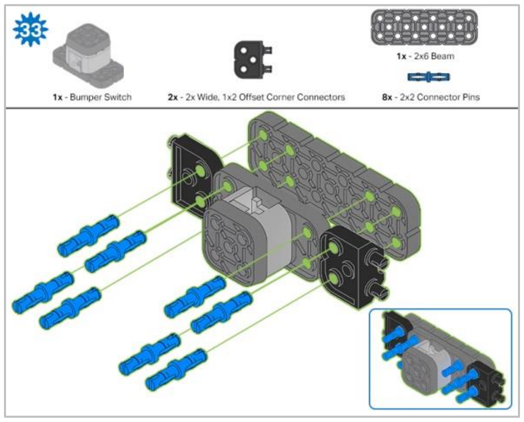




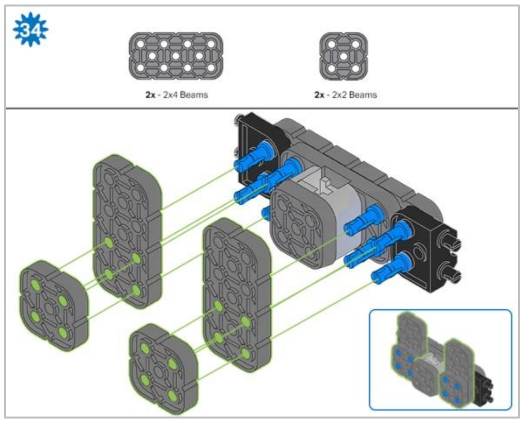


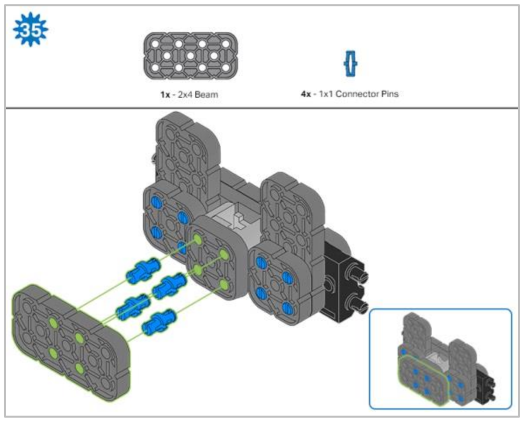
Step 31: The orange arrows mean spin the build around.

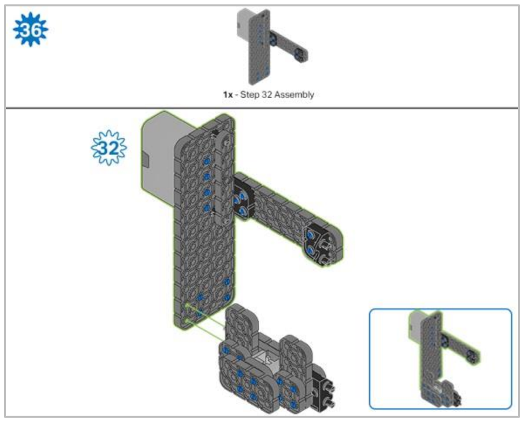




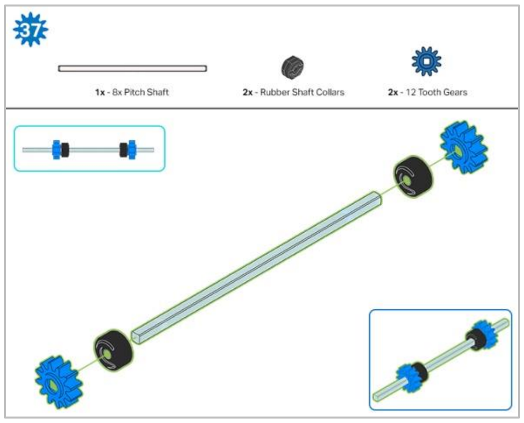
Step 33: Make sure the Bumper Switch is placed in the correct way to allow cable access.

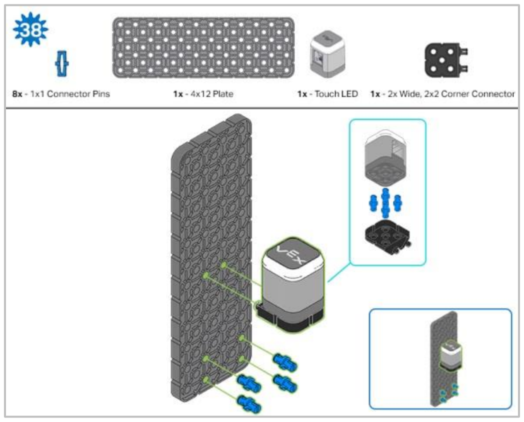




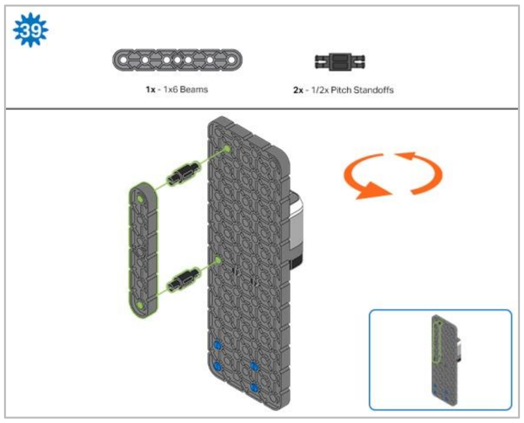


Step 36: Make sure your Smart Motor is are oriented in the correct direction (the hole for the shaft is on the bottom)

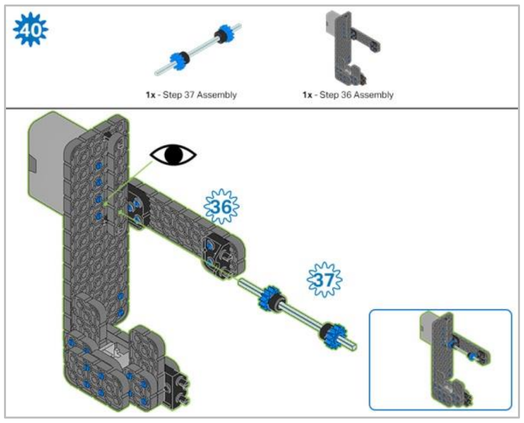




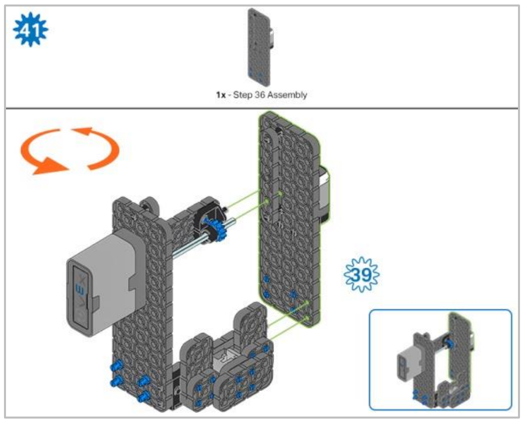
Step 38: Make sure that the Touch LED is placed in the correct way to allow cable access.



Step 39: The orange arrows mean spin the build around.



Step 40: Instead of individual parts, the completed sections of the build needed are shown in the section at the top. When adding the Step 37 Assembly, twist the pitch shaft to check for tension while turning. If it spins freely, it is not properly inserted into the motor.



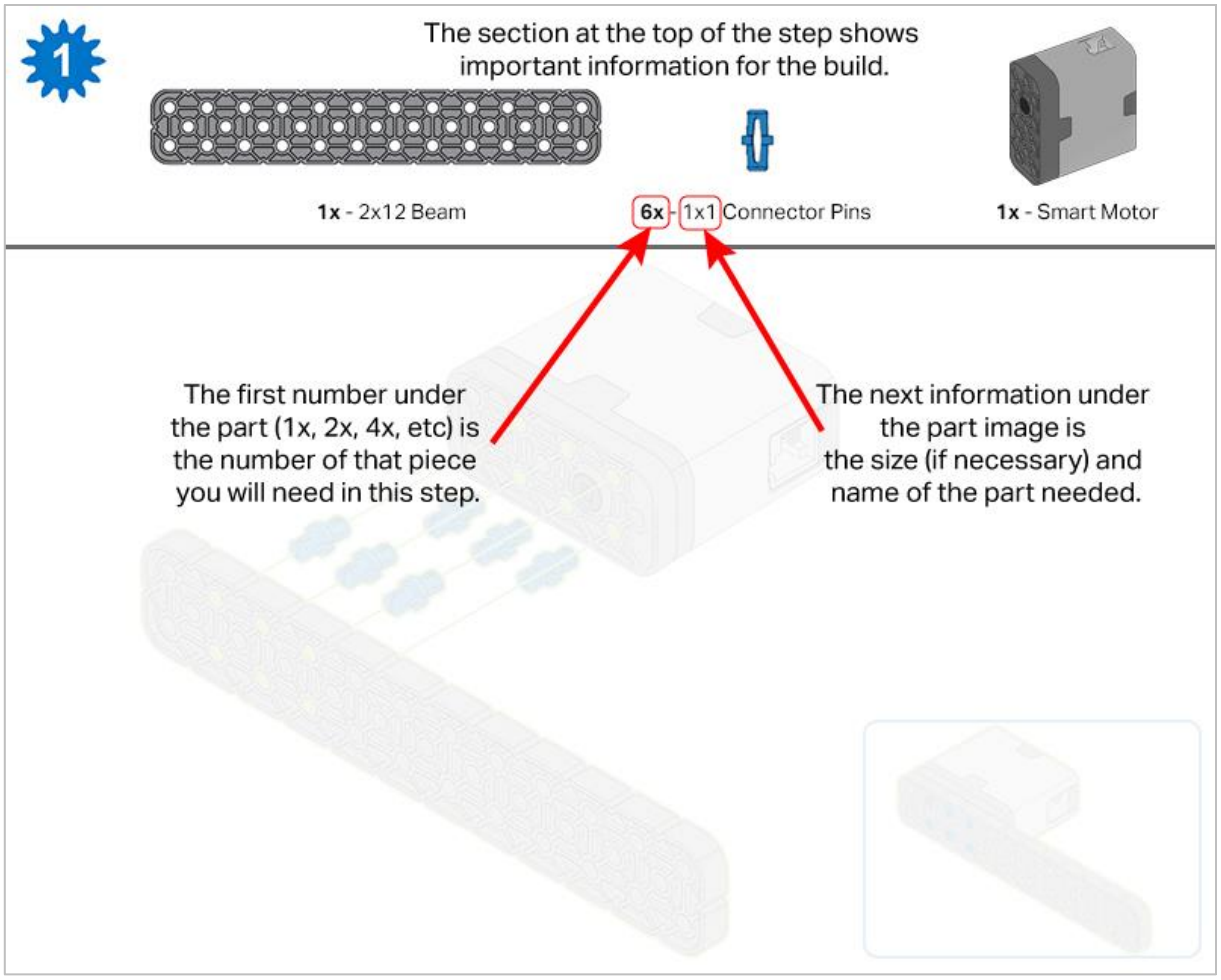
Step 41: Instead of individual parts, the completed sections of the build needed are shown in the section at the top. The orange arrows mean spin the build around.

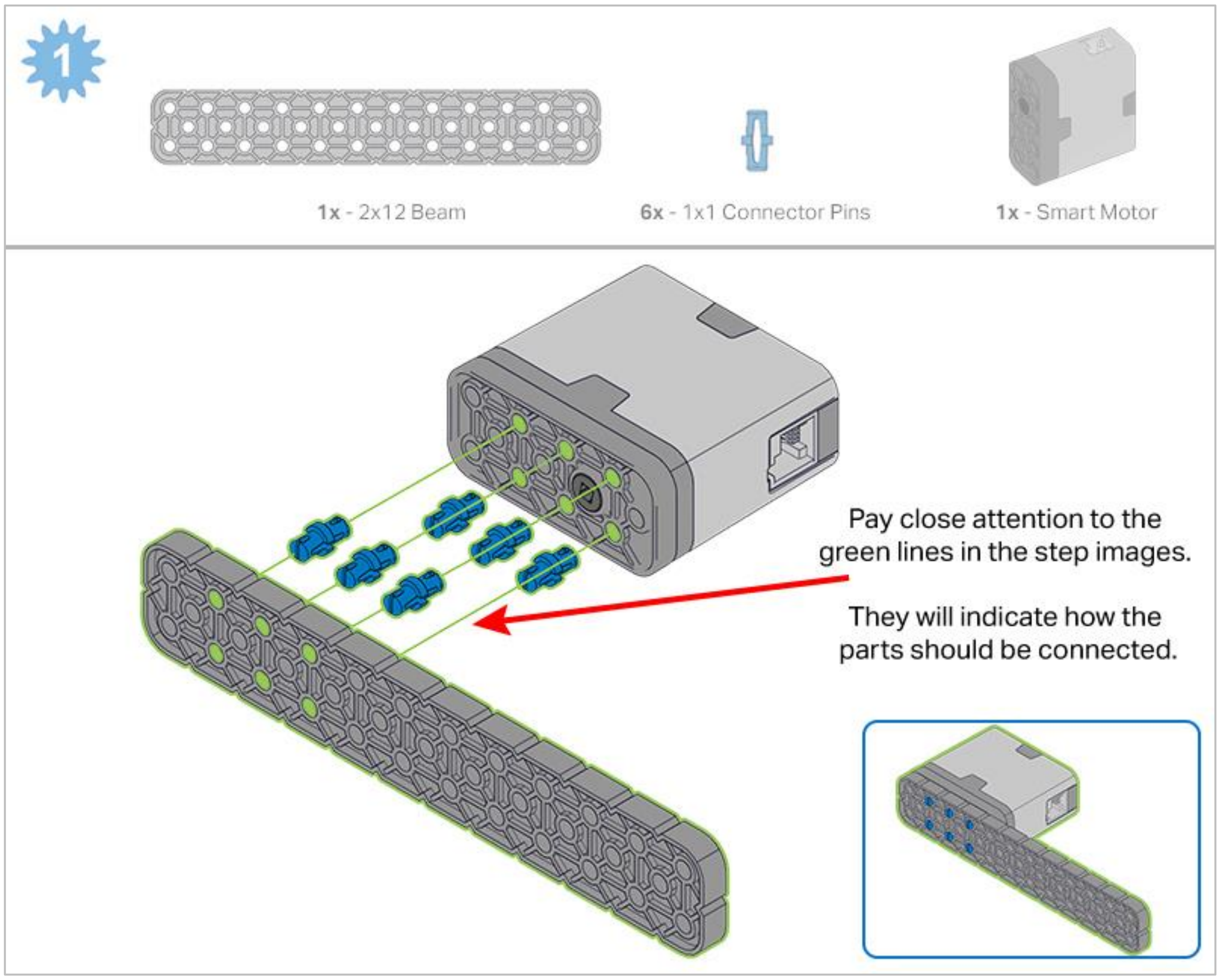
Build Instructions - Arm

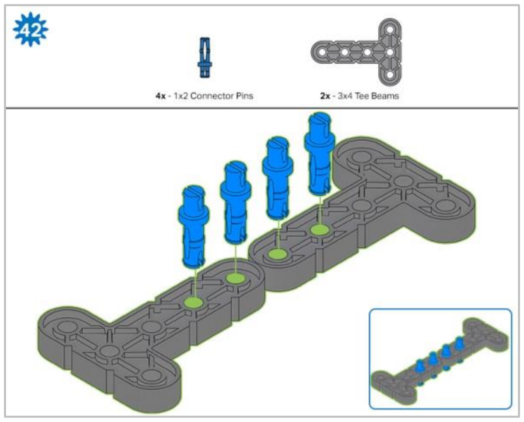
Build Instructions Summary

- **Arm Building Instructions (19 steps):**
  - Arm: steps 42 to 60
- **Building Tips for All Steps:**
  - The section at the top of the step shows important information for the build. The first number under the image of the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step. The next information under the part image is the size and description of the part needed.
  - The finished step is illustrated in the box in the lower right corner.

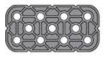
- Play close attention to the green lines in the step images. They will indicate how the parts should be connected.


1
The section at the top of the step shows important information for the build.
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
The first number under the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step.
The next information under the part image is the size (if necessary) and name of the part needed.

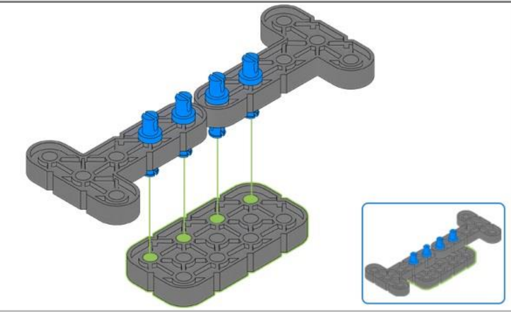

1
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
Pay close attention to the green lines in the step images.
They will indicate how the parts should be connected.


42
4x - 1x2 Connector Pins
2x - 3x4 Tee Beams

43

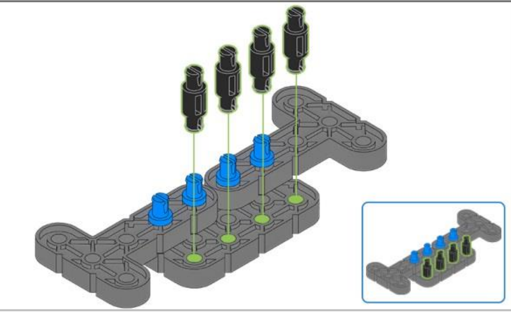


1x - 2x4 Beam



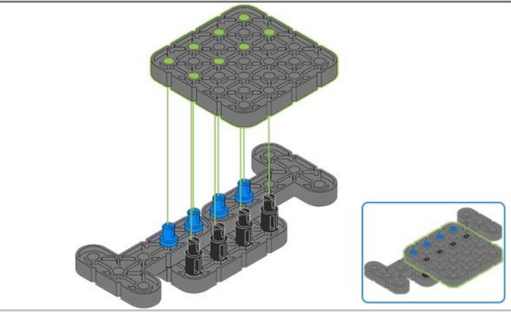
44

4x - 1/2x Pitch Standoffs



45

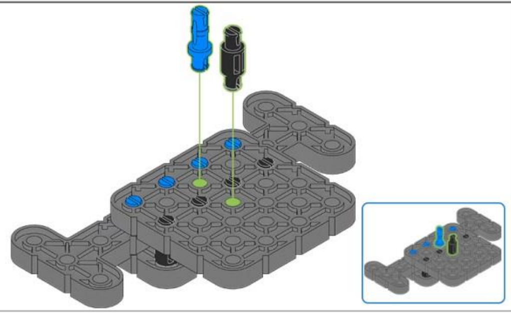
1x - 4x4 Plate

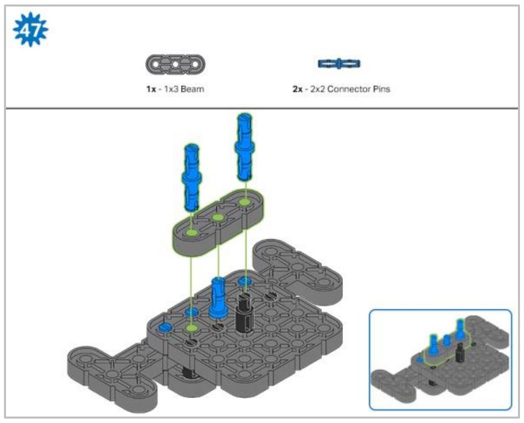


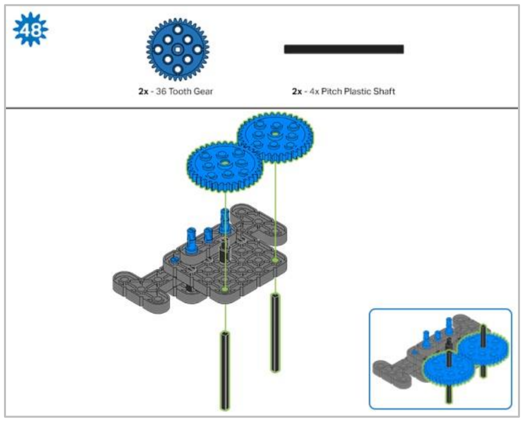
46

1x - 1x2 Connector Pin

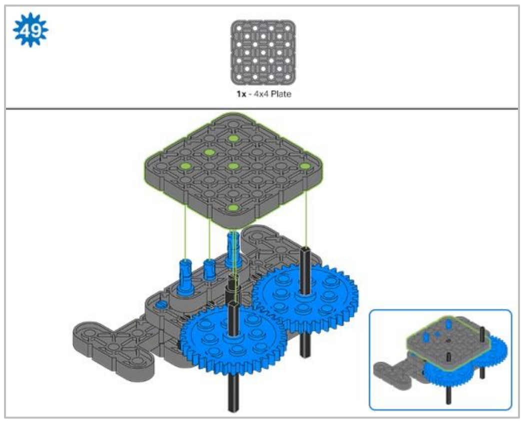
1x - 1/2x Pitch Standoff







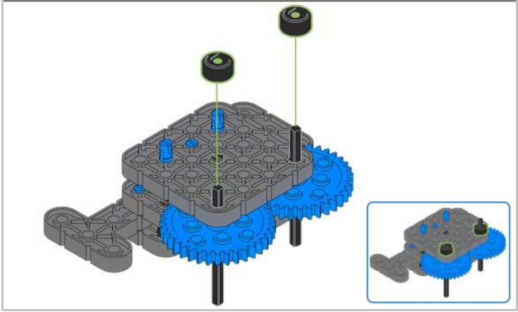
Step 48: Make sure the gears fit together properly before moving on to the next step.



Step 49: Turn one of the black shafts in the center of the gear to make sure they are together and both turn at the same time before adding the 4x4 Plate.

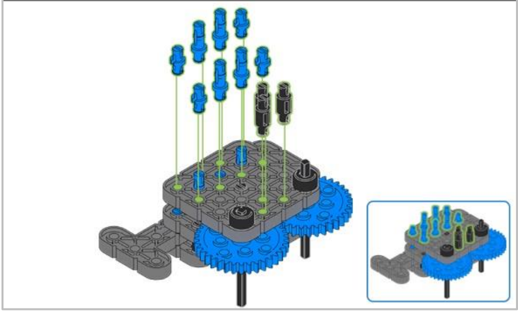
50

2x - Rubber Shaft Collars



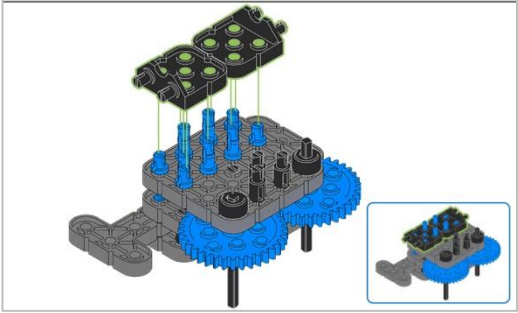
51

4x - 1x1 Connector Pins **4x** - 1x2 Connector Pins **3x** - 1/2x Pitch Standoffs



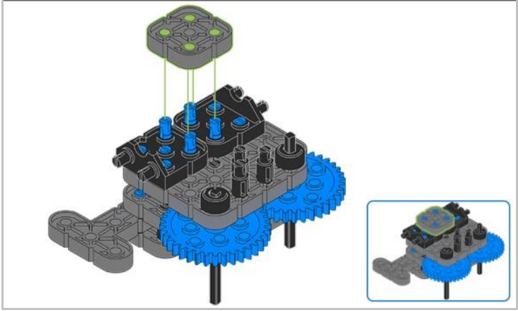
52

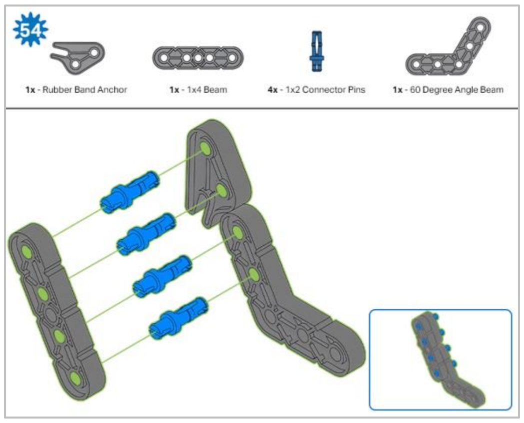
2x - 2x Wide, 2x2 Corner Connectors

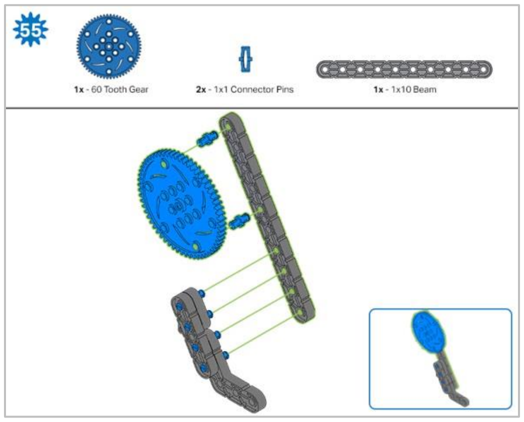


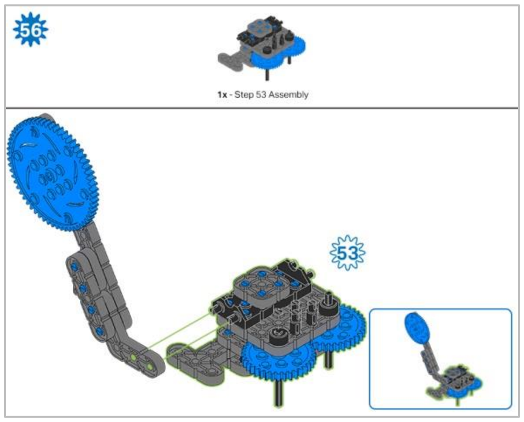
53

1x - 2x2 Beam

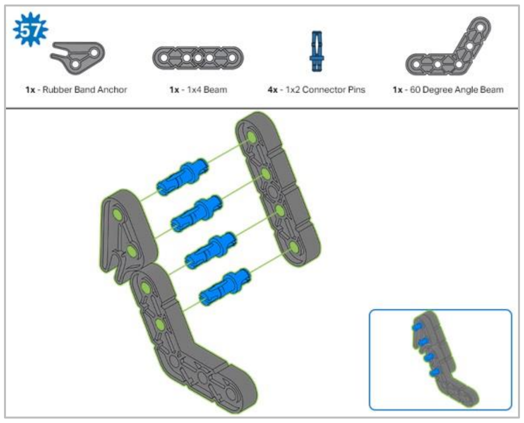


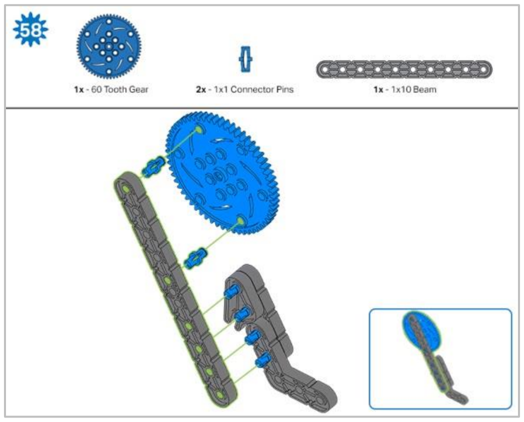


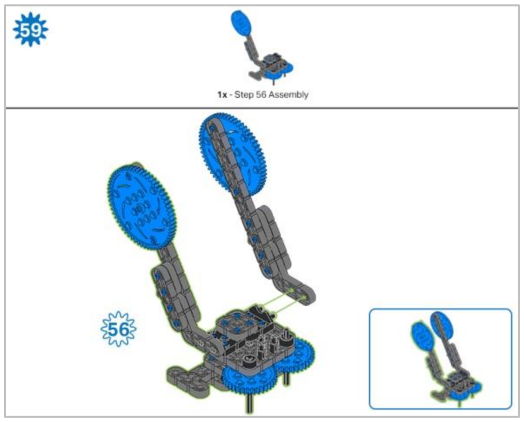




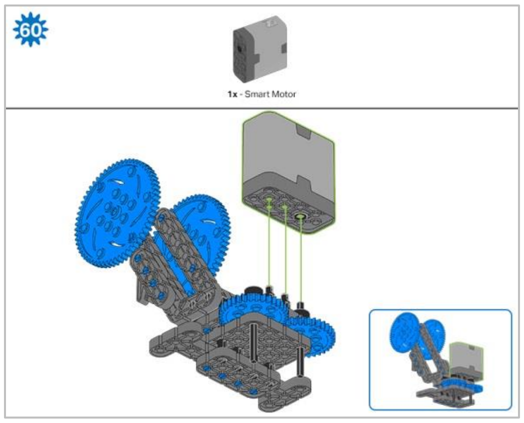
Step 56: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.







Step 59: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.



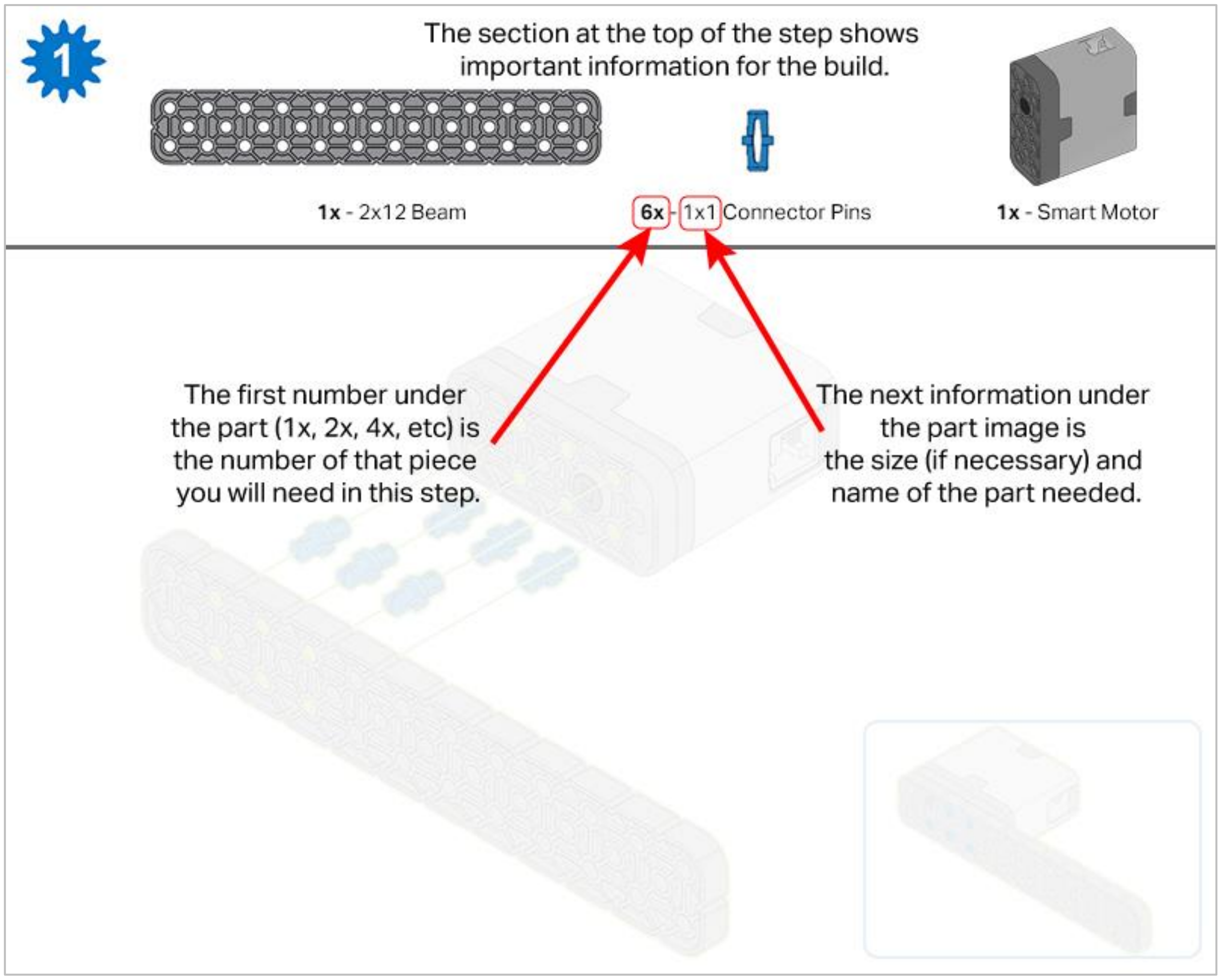
Step 60: Make sure your Smart Motor is are oriented in the correct direction (the hole for the shaft is on the right). After adding motor, turn one of the gears to check for tension while turning. If it spins freely, it is not properly inserted into the motor.

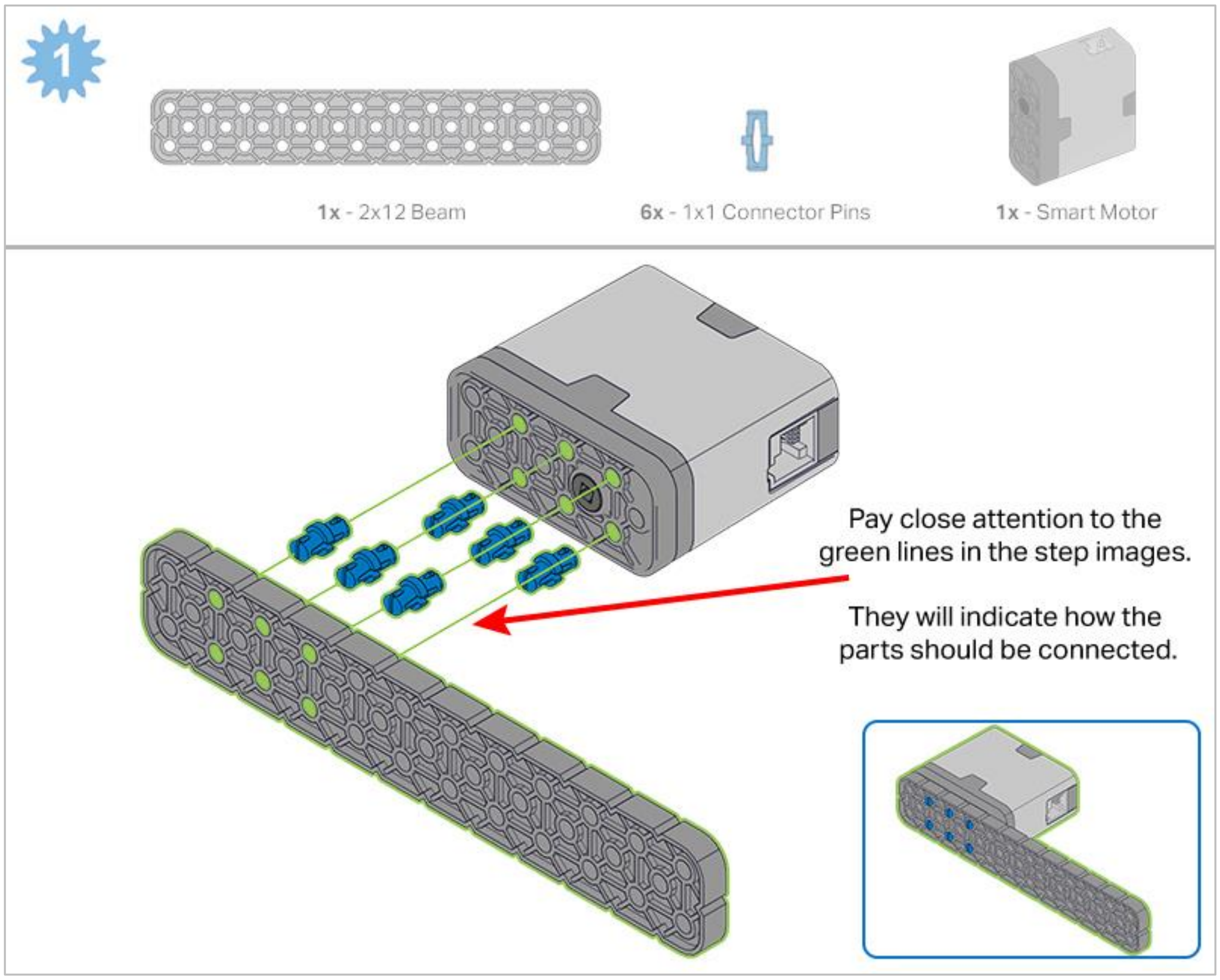
Build Instructions - Claw

Build Instructions Summary

- **Claw Building Instructions (22 steps):**
  - Claw: steps 61 to 82
- **Building Tips for All Steps:**
  - The section at the top of the step shows important information for the build. The first number under the image of the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step. The next information under the part image is the size and description of the part needed.
  - The finished step is illustrated in the box in the lower right corner.

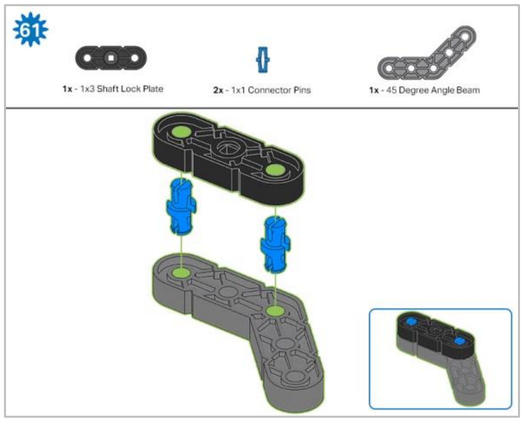
- Play close attention to the green lines in the step images. They will indicate how the parts should be connected.

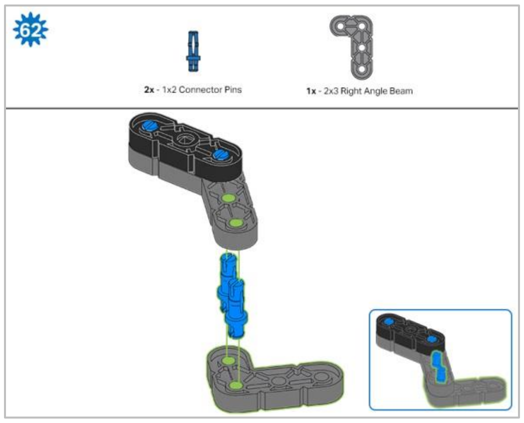

1
The section at the top of the step shows important information for the build.
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
The first number under the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step.
The next information under the part image is the size (if necessary) and name of the part needed.

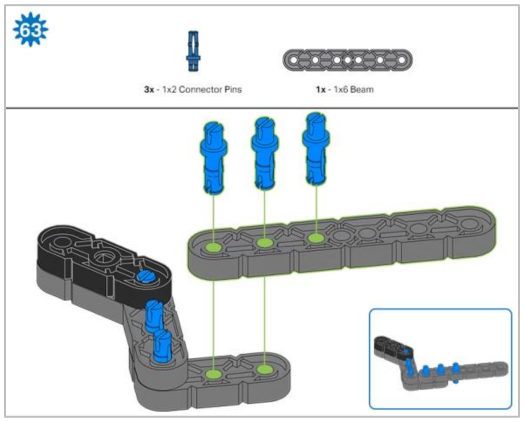


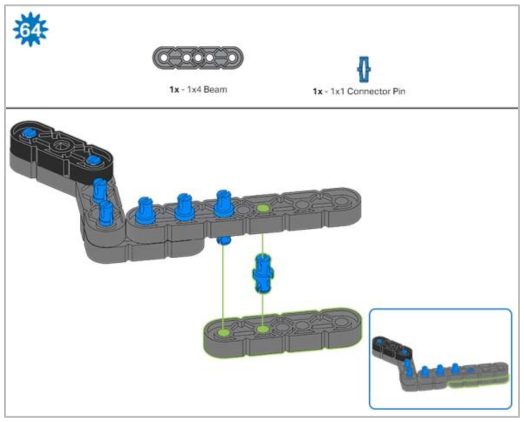
Pay close attention to the green lines in the step images.

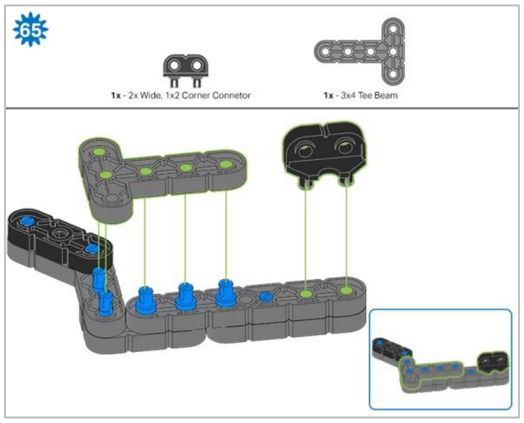
They will indicate how the parts should be connected.

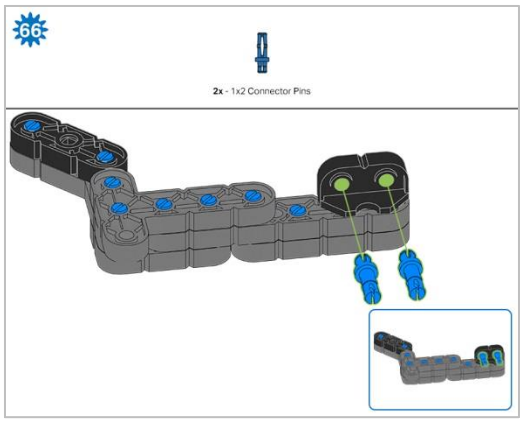


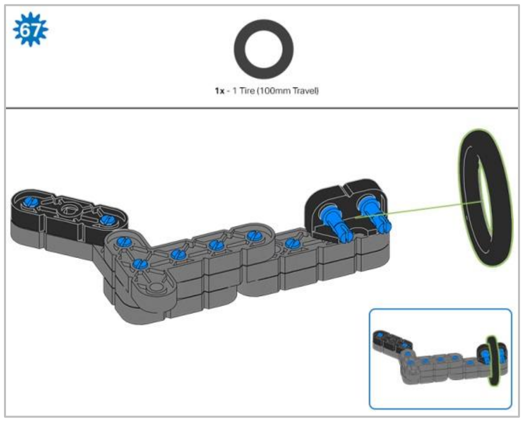

62
2x - 1x2 Connector Pins
1x - 2x3 Right Angle Beam


63
3x - 1x2 Connector Pins
1x - 1x6 Beam

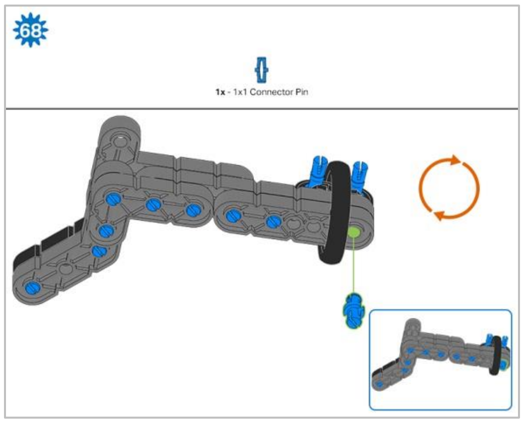

64
1x - 1x4 Beam
1x - 1x1 Connector Pin


65
1x - 2x Wide, 1x2 Corner Connetor
1x - 3x4 Tee Beam

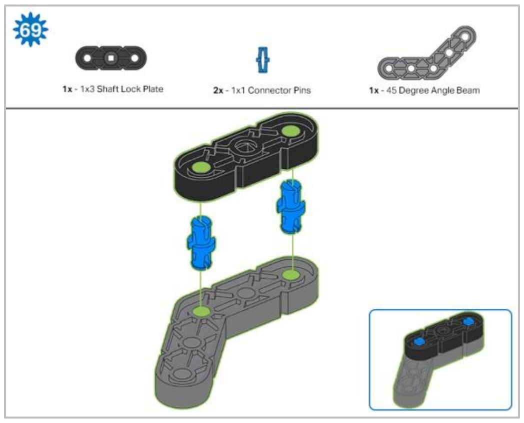


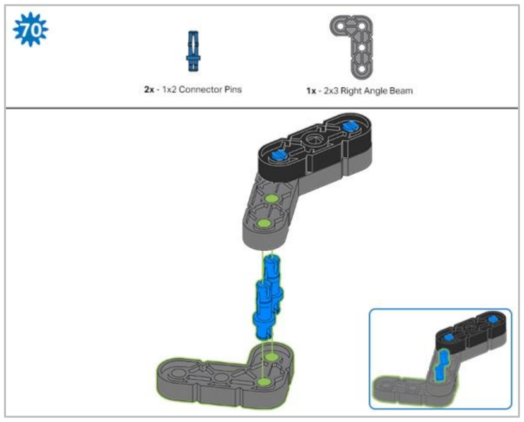


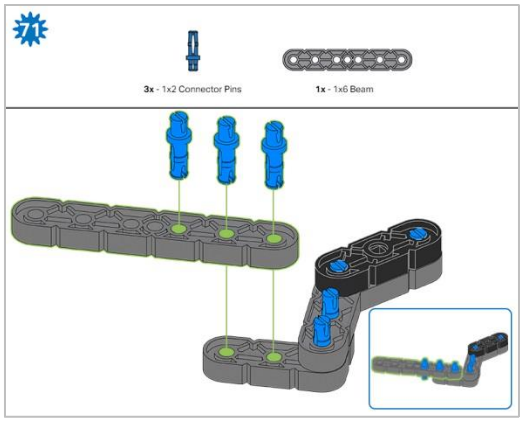
Step 67: Make sure that the 100mm Travel Tire fits snugly in the grove of the 2x Wide, ½ Corner Connector.

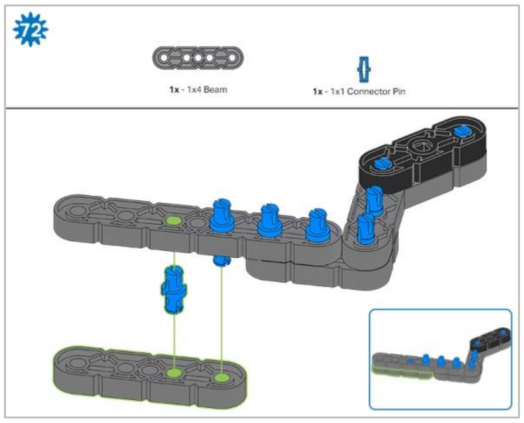


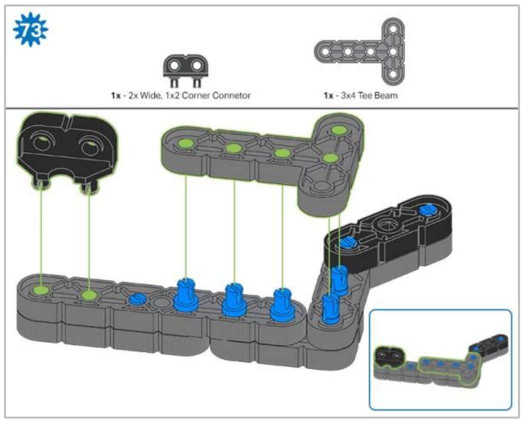
Step 68: The orange arrows mean spin the build around.

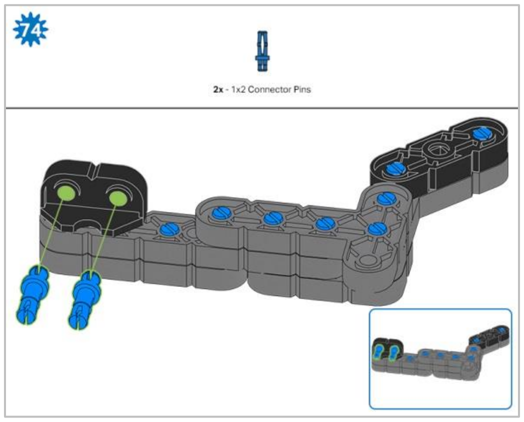

69
1x - 1x3 Shaft Lock Plate
2x - 1x1 Connector Pins
1x - 45 Degree Angle Beam

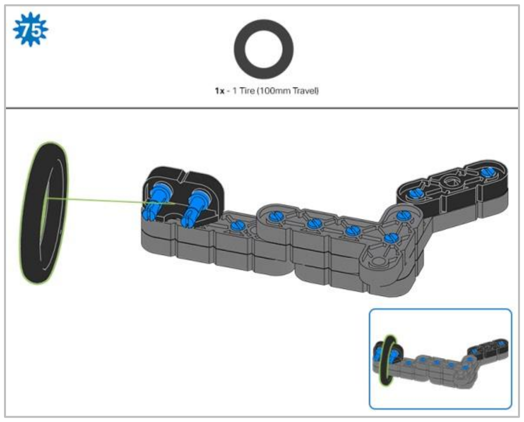

70
2x - 1x2 Connector Pins
1x - 2x3 Right Angle Beam


71
3x - 1x2 Connector Pins
1x - 1x6 Beam

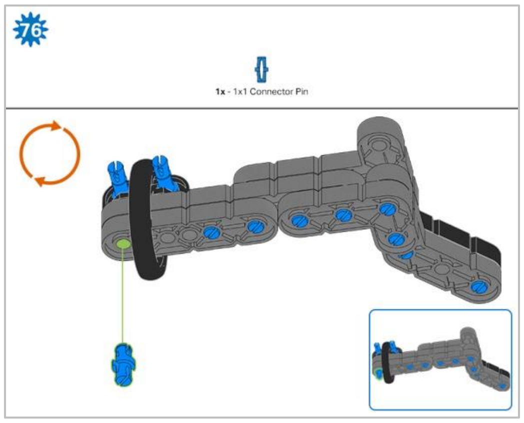

72
1x - 1x4 Beam
1x - 1x1 Connector Pin



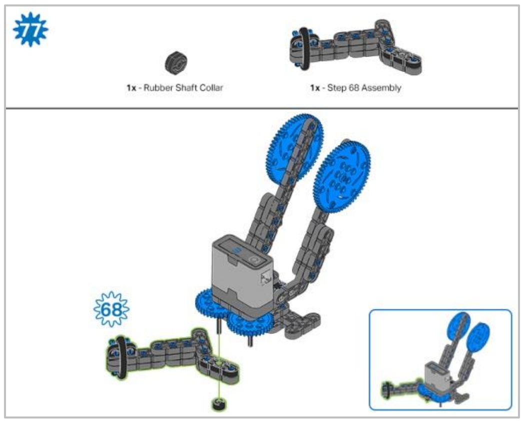




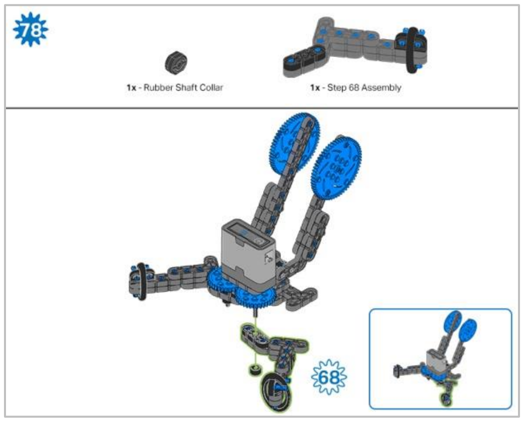
Step 75: Make sure that the 100mm Travel Tire fits snugly in the grove of the 2x Wide, ½ Corner Connector.



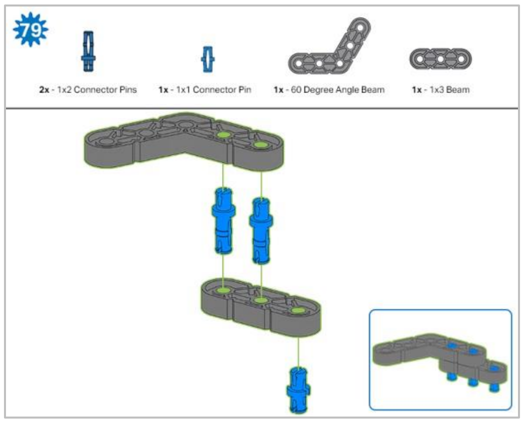
Step 76: The orange arrows mean spin the build around.

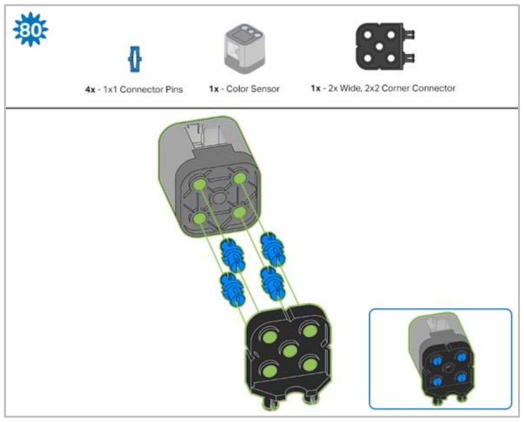


Step 77: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.

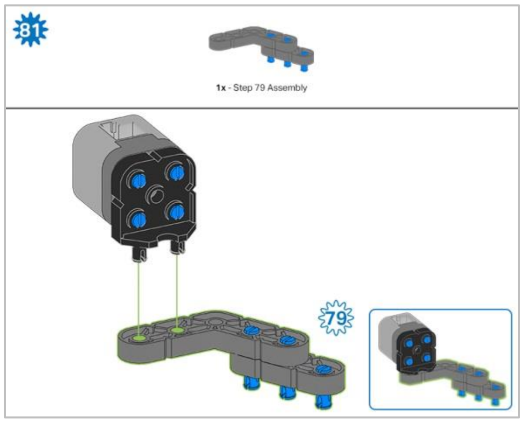


Step 78: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.





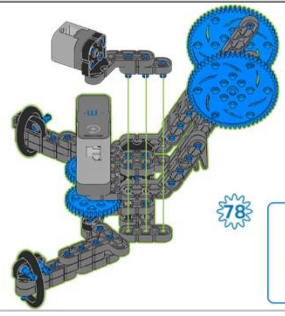
Step 80: The sensor being attached is the Color Sensor. Ensure the sensor is placed in the correct way to allow cable access.



Step 81: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.

82

1x - Step 78 Assembly



78

Build Instructions - Assembly and Wiring

Build Instructions Summary

- **Assembly and Wiring (11 steps):**
  - Final Assembly: steps 83 to 93
  - The group is also responsible for making sure the sensors and motors are attached to the correct ports using the designated Smart Cables.

    − Port 1: Left Wheel

    − Port 2: Touch LED

    − Port 3: Color Sensor

    − Port 4: Gyro Sensor

    − Port 6: Right Wheel

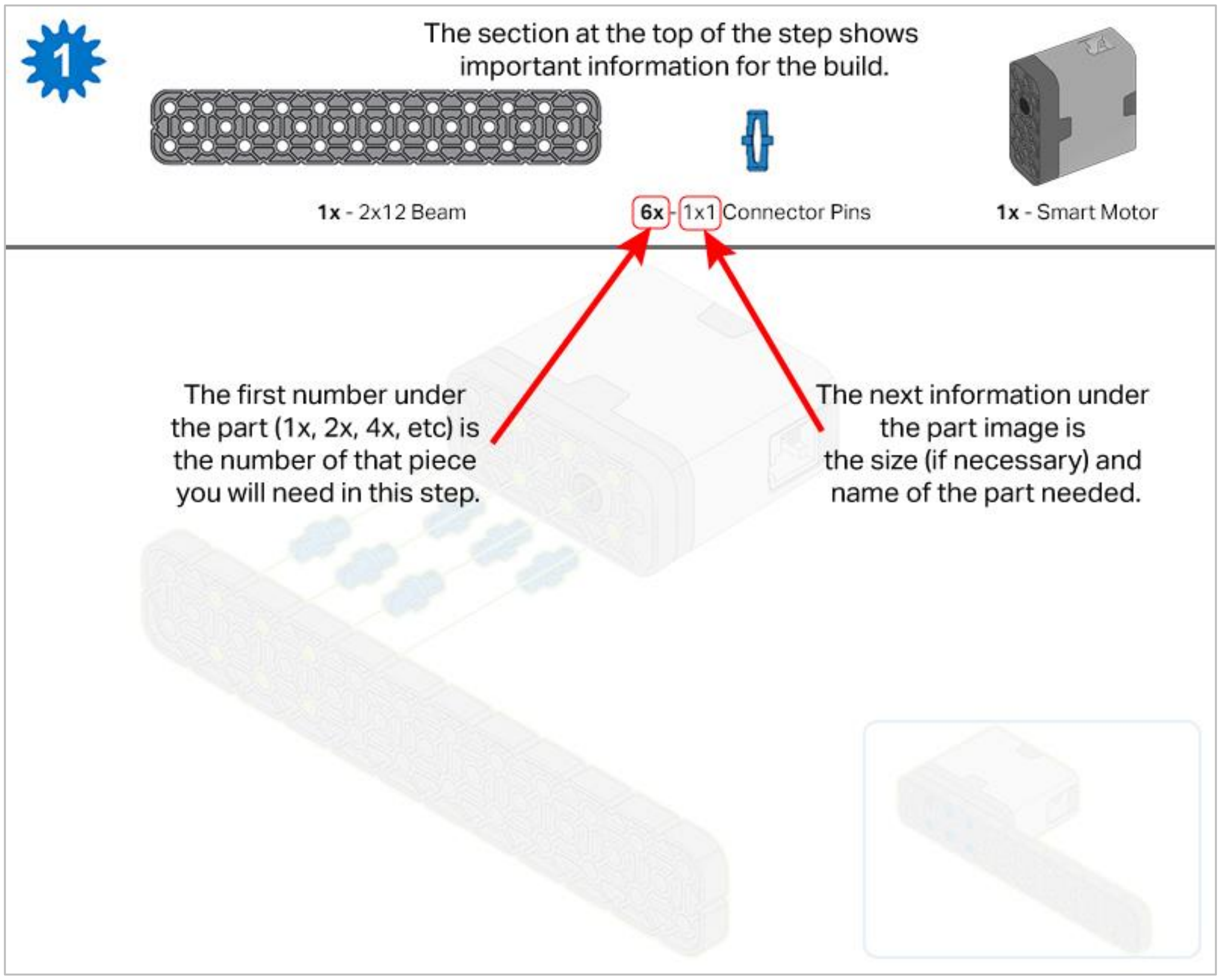
    − Port 7: Distance Sensor

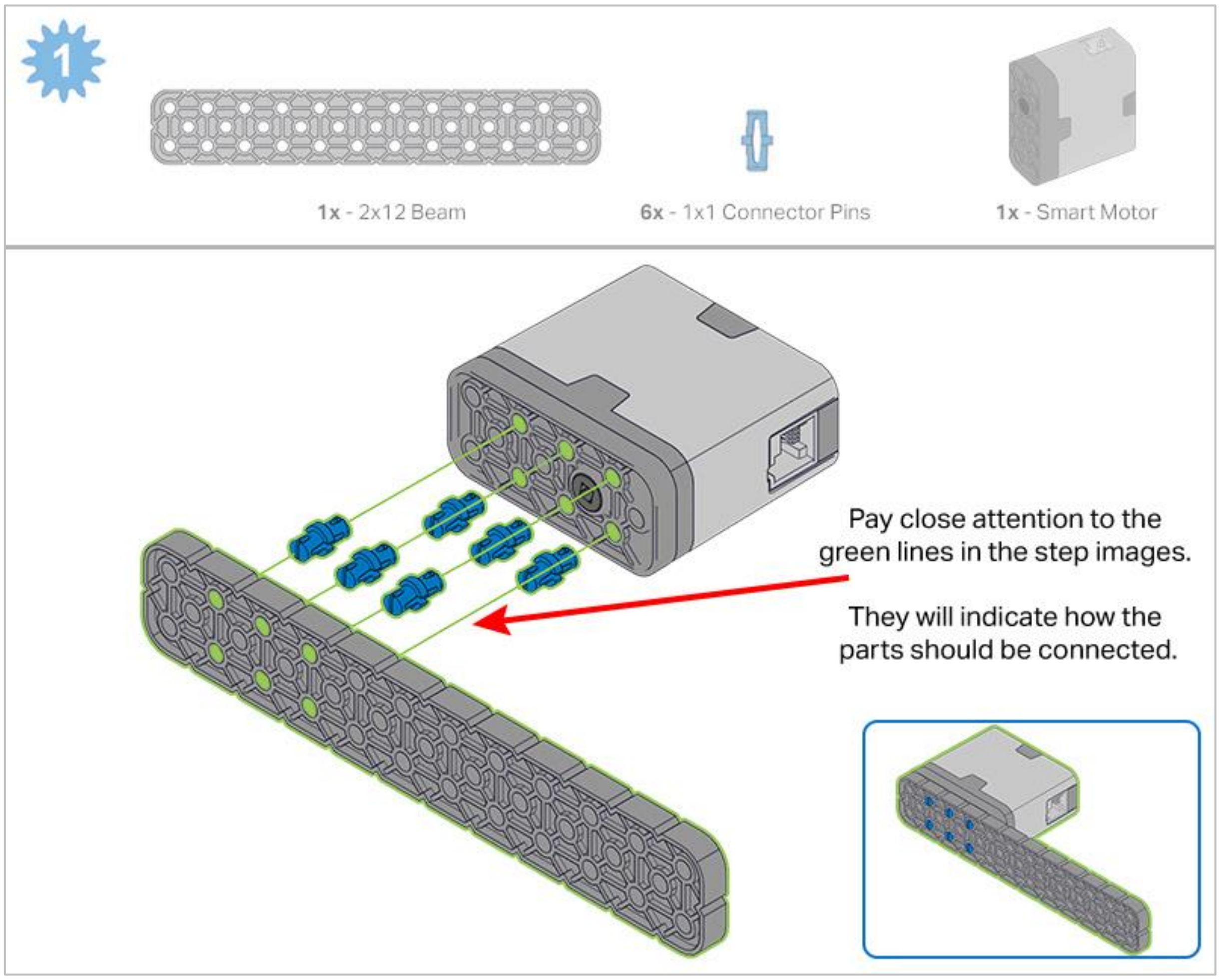
    − Port 8: Bumper Switch

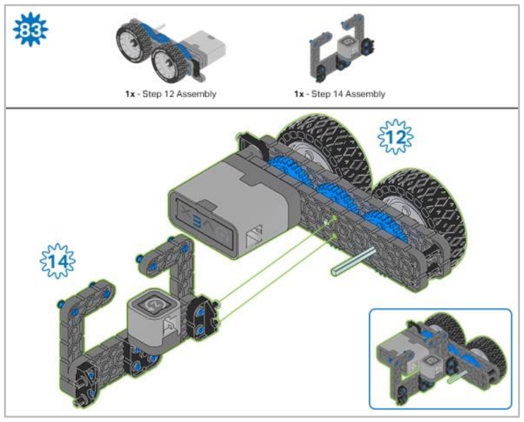
    − Port 10: Arm Motor

    − Port 11: Claw Motor
- **Building Tips for All Steps:**
  - The section at the top of the step shows important information for the build. The first number under the image of the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step. The next information under the part image is the size and description of the part needed.
  - The finished step is illustrated in the box in the lower right corner.
  - Play close attention to the green lines in the step images. They will indicate how the parts should be connected.
- **Building Tips for Steps 87-89:**

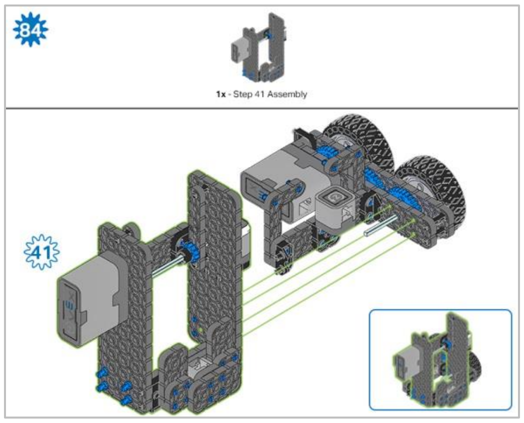
- The solid green numbers represent the numbered port the cable will be connected into. The outlined green number indicates the sensor that cable will connect into. Use the indicated Smart Cable for each sensor or motor. When attaching the Smart Cables, make sure they are tucked away so as to not block the Smart Sensors or interfere with the Clawbot's movement.


1
The section at the top of the step shows important information for the build.
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
The first number under the part (1x, 2x, 4x, etc) is the number of that piece you will need in this step.
The next information under the part image is the size (if necessary) and name of the part needed.

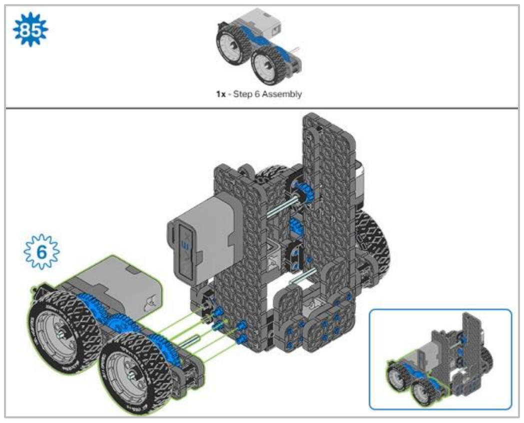

1
1x - 2x12 Beam
6x - 1x1 Connector Pins
1x - Smart Motor
Pay close attention to the green lines in the step images.
They will indicate how the parts should be connected.



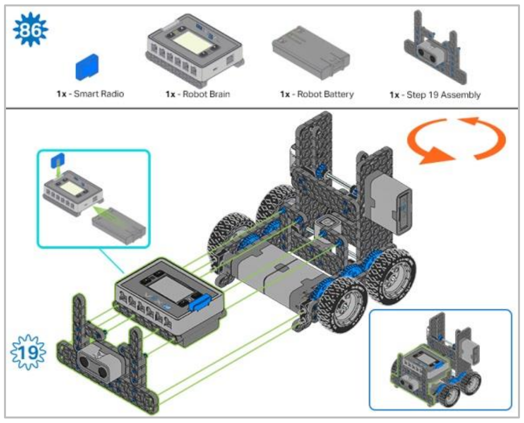
Step 83: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.



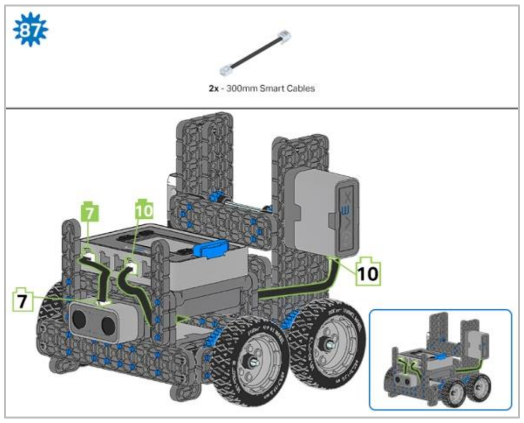
Step 84: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.



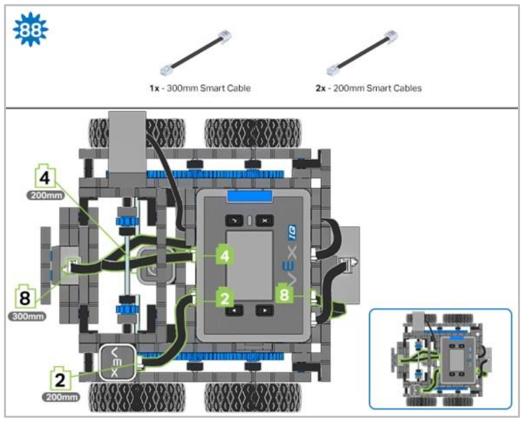
Step 85: Instead of individual parts, the completed sections of the build needed are shown in the section at the top.

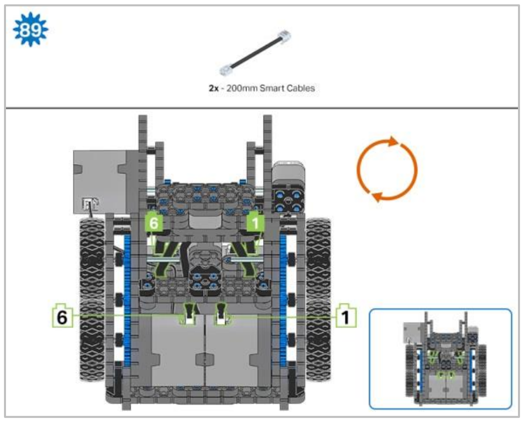


Step 86: Make sure the Smart Radio is pushed in securely. Make sure the Robot Battery is oriented the correct way before inserting. The orange arrows mean spin the build around.

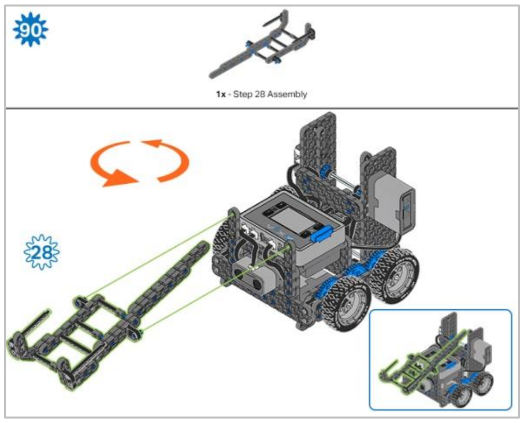


Step 87: The Smart cable for the Arm Motor can be tucked under the Brain and plugged into the correct port (port 10).

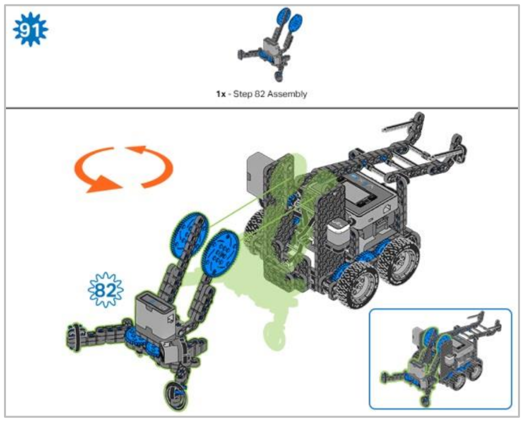




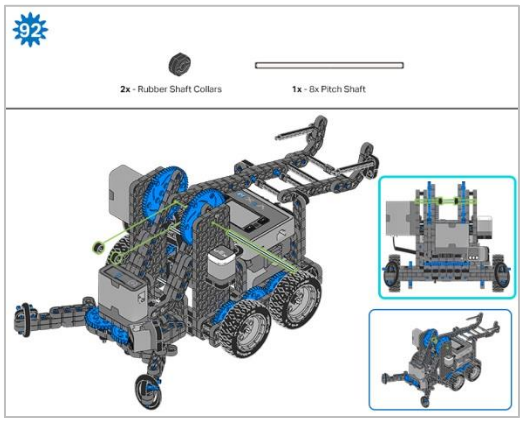
Step 89: The orange arrows mean spin the build around.



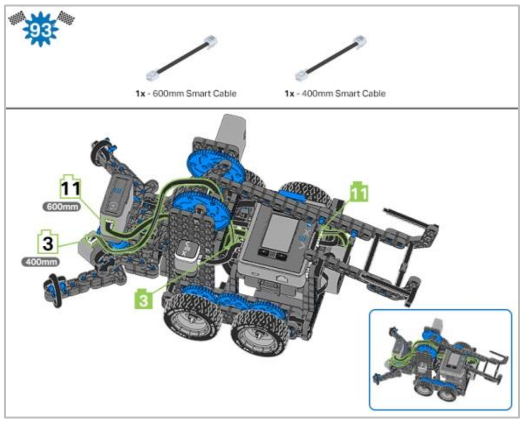
Step 90: Instead of individual parts, the completed sections of the build needed are shown in the section at the top. The orange arrows mean spin the build around.



Step 91: Instead of individual parts, the completed sections of the build needed are shown in the section at the top. The orange arrows mean spin the build around.



Step 92: When adding the 8x Pitch Shaft, twist the pitch shaft to check for tension while turning. If it spins freely, it is not properly inserted into the gears.



Step 93: The solid green numbers represent the numbered port the cable will be connected into. The outlined green number indicates the sensor that cable will connect into. Use the indicated Smart Cable for each sensor. When attaching the Smart Cables, make sure they are tucked away so as to not block the Smart Sensors or interfere with the Clawbot's movement.

Exploration

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

How would the speed of the arm change if the 60 tooth gears in Steps 55 and 58 were changed to smaller 36 tooth gears?

- For help with this question, look at the rotational speed (RPM) of the 12 tooth gears (from Build Step 37) and compare it to the rotational speed of the 60 tooth gear by gently moving the arm up and down.
  - Observe how many times the 12 tooth gear rotates compared to the 60 tooth gear. How do you think this will change when the 60 tooth gear is replaced by a 36 tooth gear?
  - Be sure to justify your answer with data from your observation.

SPARK
PLAY

Test your build, observe how it functions,

and fuel your logic and reasoning skills

through imaginative, creative play.

Decision Making

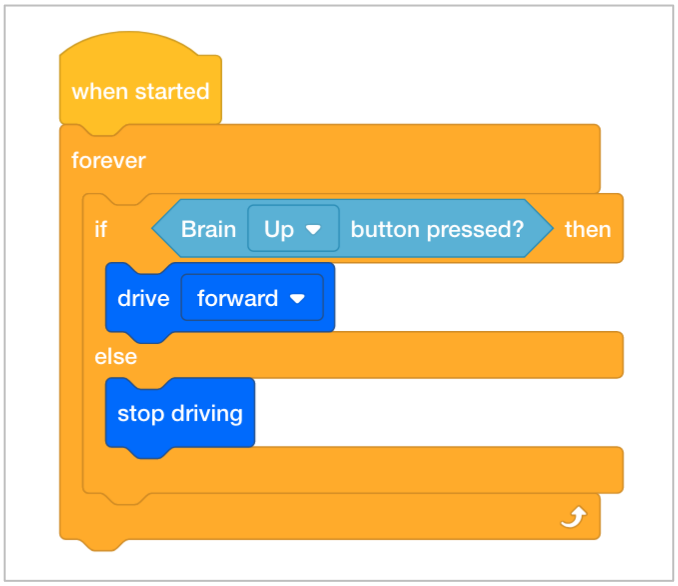
TRUE and FALSE paths

Decision Making

At their most basic level, programs are written to accomplish simple sequences of behavior. For example, you might want your robot to drive forward and also make some turns to reach a destination. But, what if you want your robot to wait for the right time to start driving forward and complete its route? That would require programming with conditional statements. You would use a conditional statement to define what the "right time to start" is within your project. Maybe the "right time" is after a button is pressed or when a sensor detects a specific level and then it starts driving. When you watch the robot's behavior, it will seem like it is deciding when to start driving but it's because you set the condition for when driving should start.

Conditional statements are powerful programming statements that use a boolean (TRUE or FALSE) condition and allow you to develop projects that have the robot behave differently depending on what it senses.

In the following example, if the Brain Up button is pressed (TRUE) the robot will drive forward. If the Brain Up button is not pressed (FALSE) the robot will stop driving. This shows the robot only driving forward when the Brain Up button is pressed, otherwise the robot stops.



```
when started
forever
    if Brain Up ▾ button pressed? then
        drive forward ▾
    else
        stop driving
```

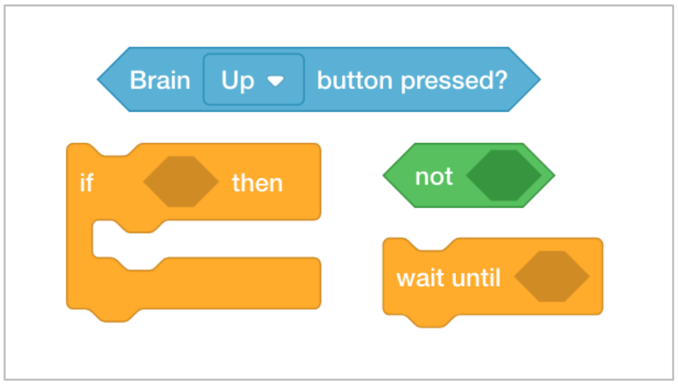
Programming with Conditionals in VEXcode IQ Blocks

Hardware/Software Required:

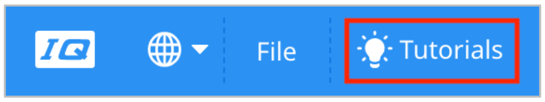
Amount	**Hardware/Software**
1	VEX IQ Super Kit (with up-to-date firmware)
1	VEXcode IQ Blocks (latest version, Windows, macOS, Chromebook, iPad)
1	Engineering Notebook
1	Clawbot (Drivetrain) Template

The Clawbot is ready to make decisions!

This activity will give you the tools to program your robot with conditional behaviors. The *if then* block is the main focus within the activity but other Sensing, Control, and Operator blocks are also used.



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

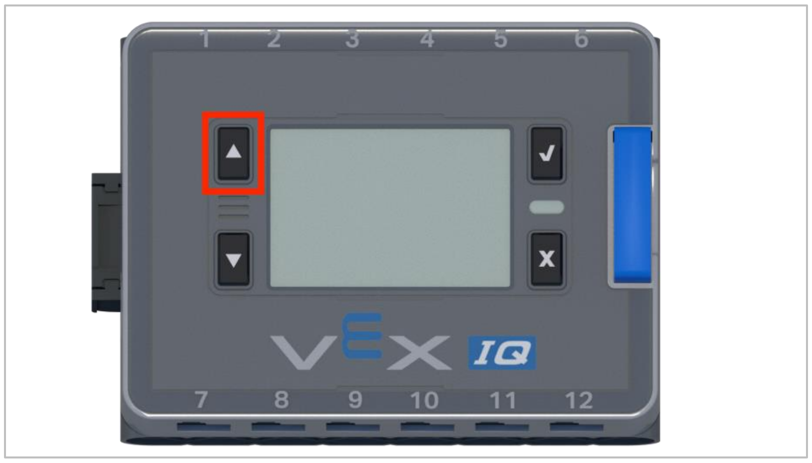


1. Let's start with an understanding of loops and conditional statements.

Before you bein programming with conditionals, first watch the Using Loops and If-Then-Else tutorial videos in VEXcode IQ Blocks.

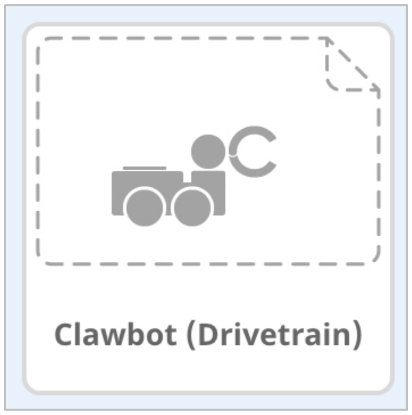
2. Programming with conditionals.

We want to build a project that will raise the arm if the condition of the Brain Up button being pressed is true.

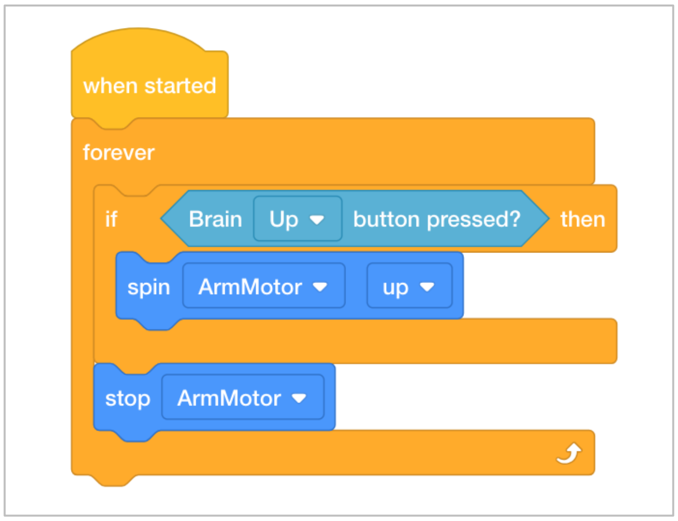


If the condition is false, the Arm Motor will stop. Let's explore building a project that uses a *forever* block and the *if then* conditional block to program the arm.

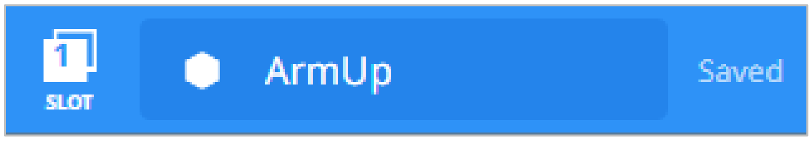
- Open the **Clawbot (Drivetrain) Template. For help, view the** Using Example Projects and Templates **tutorial video.**



- Build the project below.


```
when started
forever
    if <Brain [Up] button pressed?> then
        spin [ArmMotor] [up]
    stop [ArmMotor]
```

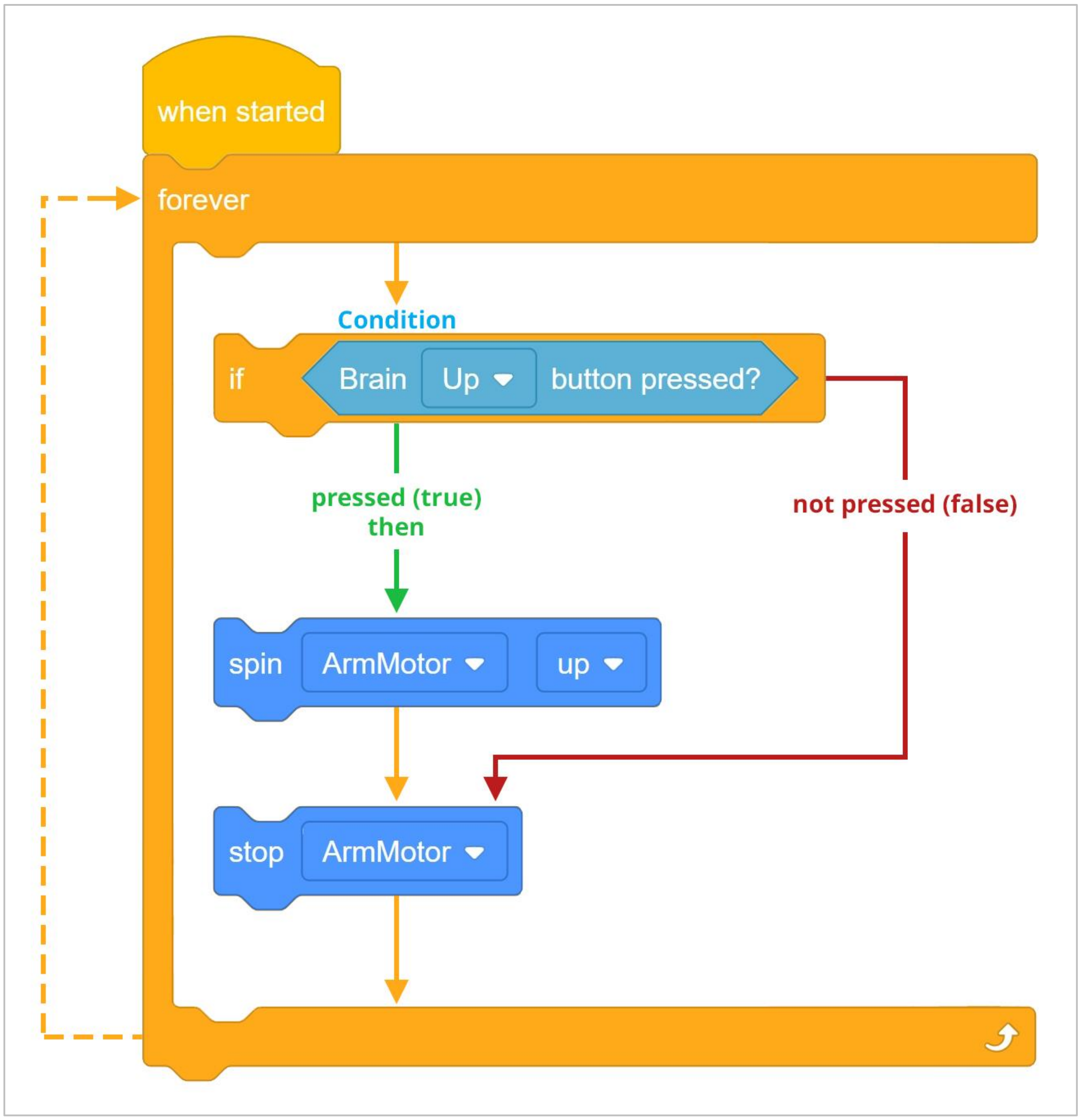
- Save the project as ArmUp. For help, watch the **Naming and Saving Your Project** tutorial video (Windows, Mac, iPad, Chrome) in VEXcode IQ Blocks.



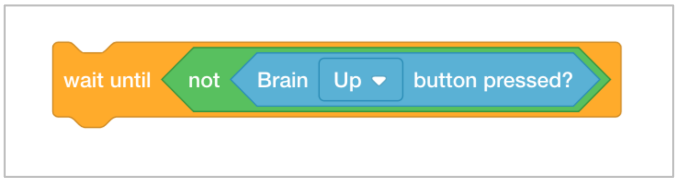
- Check to make sure the project name **ArmUp** is now in the window in the center of the toolbar.
- Predict what the project will have the Clawbot do in your engineering notebook. Explain both the user's and the Clawbot's behaviors.
- Test to see if your prediction of what the project has the Clawbot do is correct.
  - Download the project to Slot 1 on the Clawbot, and then run it.
  - For help downloading and running a project, see the tutorial in VEXcode IQ Blocks that explains how to Download and Run a Project.
- Check your explanations of the project and add notes to correct them as needed.

3. Understanding the *wait until* block.

In the previous step, the project did not raise the arm successfully. View the following flowchart that explains the project flow. Notice that if the Brain Up button is pressed, the flow of the project moves so quickly that the project will move to the next block, which is the *stop motor* block.



Thus, the project needs a *wait until* block that tells the Arm Motor to keep spinning until the Brain Up button is released.

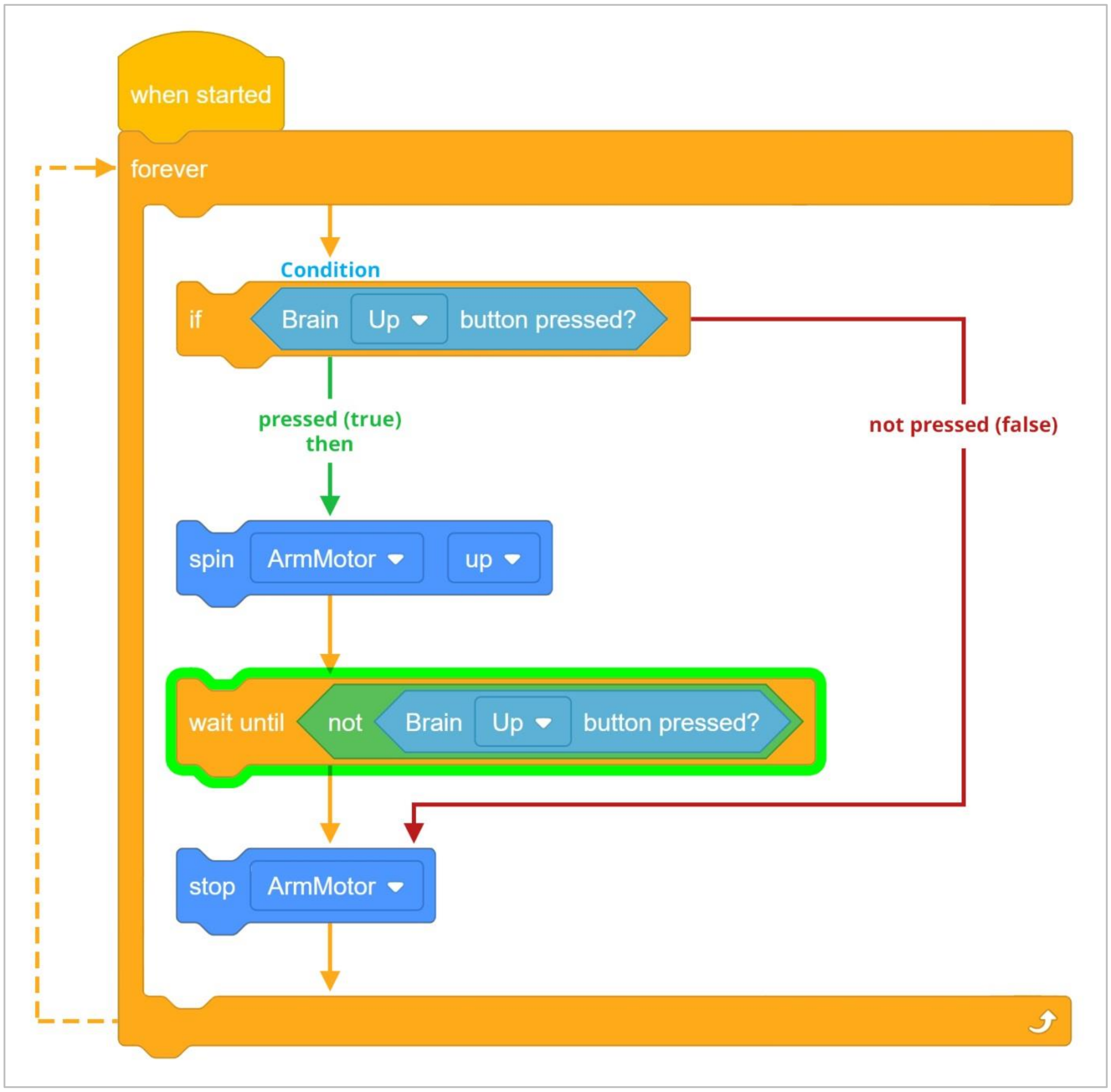


The *wait until* block is necessary because of the speed of the project's flow. If it was not there, the project would move to the next block before the Arm Motor ever had time to respond. Thus, the blocks would flow down to the *stop motor* block and then start back at the top of the stack because of the *forever* block that repeats all blocks contained inside of it.

Let's explore changing the project by adding a *wait until* block. The Arm Motor will now continue to spin until the Brain Up button is released. Once the Brain Up button is released, the project will continue to the next block, which is the *stop motor* block.

The project will now first check the condition if the Brain Up button is being pressed. If the Brain Up button is being pressed (TRUE), then the arm will spin up until the Brain Up button is released. Once the Brain Up button is released, the project will move to the *stop motor* block before returning to the top of the stack to begin again because of the *forever* block.

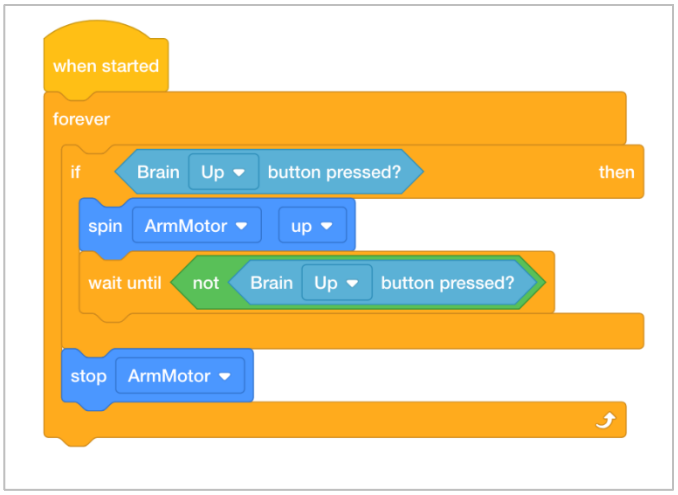
If the Brain Up button is not pressed (FALSE), then the project will move to the *stop motor* block before returning to the top of the stack to begin again because of the *forever* block and the arm will never spin.



4. Adding the *wait until* block.

Let's add the *wait until* block:

- Add the *wait until* block to your ArmUp project so that your project looks like the following:



```
when started
forever
    if < Brain Up button pressed? > then
        spin ArmMotor up
        wait until < not < Brain Up button pressed? > >
    stop ArmMotor
```

- Save the project as ArmUp2. For help, watch the **Naming and Saving Your Project** tutorial video (Windows, Mac, iPad, Chrome) in VEXcode IQ Blocks.

- Download the project to Slot 2 on the Clawbot, and then run it.
- For help downloading and running a project, see the tutorial in VEXcode IQ Blocks that explains how to Download and Run a Project.
- Test to see if the arm will now spin up when the Brain Up button is pressed.
- Verify that when the Brain Up button is not pressed (released) that the Arm Motor stops.
- Write your observations of how the Clawbot behaved before and after adding the *wait until* block to your project in your engineering notebook.

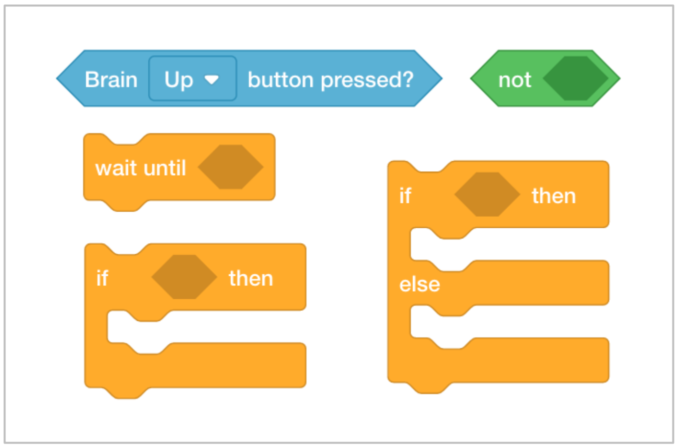
Programming Two Buttons

Hardware/Software Required:

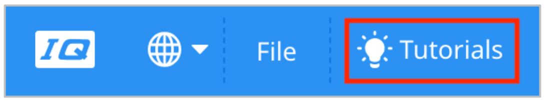
Amount	Hardware/Software
1	VEX IQ Super Kit (with up-to-date firmware)
1	VEXcode IQ Blocks (latest version, Windows, macOS, Chromebook, iPad)
1	Engineering Notebook
1	Clawbot (Drivetrain) Template

The Clawbot arm is ready to move up and down!

This activity will give you the tools to program your robot with conditional behaviors. The *if then else* block is the main focus within the activity but other Sensing, Control, and Operator blocks are also used.

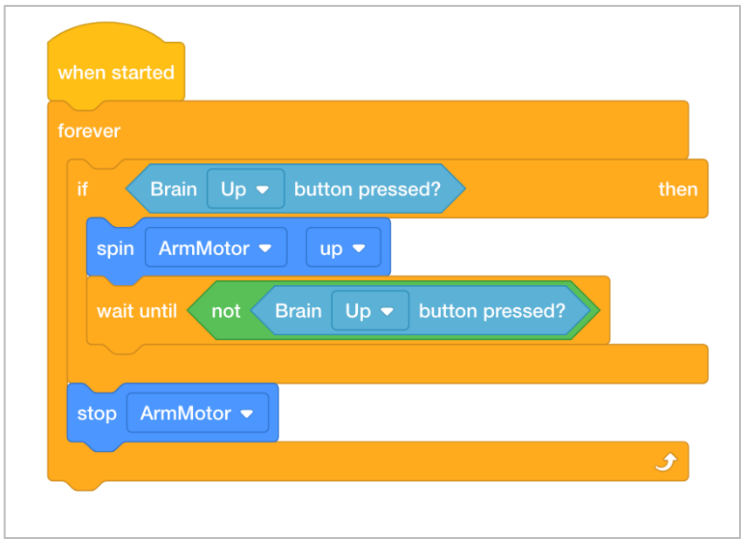


You can use the Help information inside of VEXcode IQ Blocks to learn about the blocks. For guidance in using the Help feature, see the Using Help tutorial.



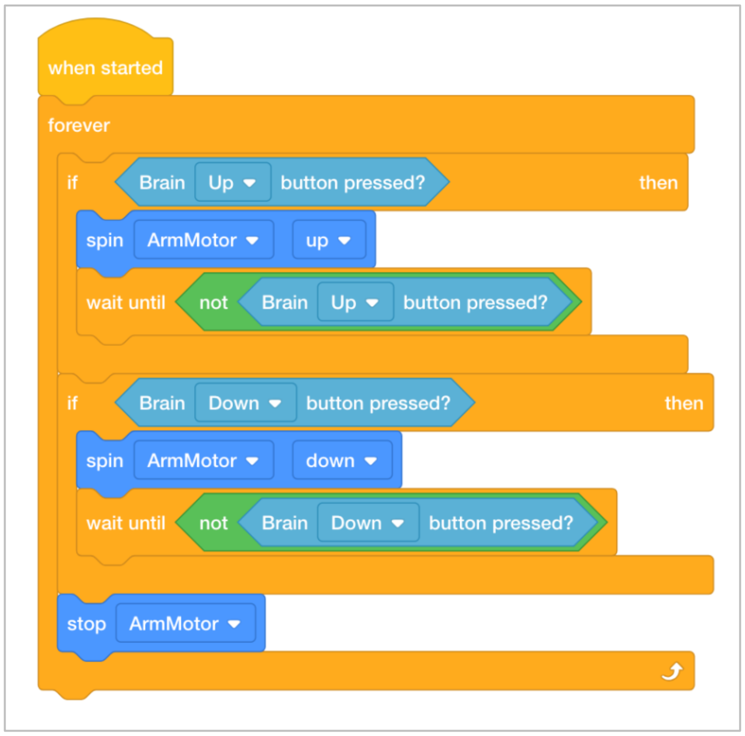
1. Programming to move the arm down.

In the previous page, you programmed the Clawbot's arm to spin up when the Brain Up button was pressed. But, what about also lowering the arm? Let's first revisit the previous ArmUp2 project.



```
when started
forever
    if <Brain Up button pressed?> then
        spin ArmMotor up
        wait until <not <Brain Up button pressed?>>
    stop ArmMotor
```

Would it be possible to use the same *if then* block for spinning the arm up to also spin the arm down? Let's try it! Build the project below by editing your already existing ArmUp2 project.



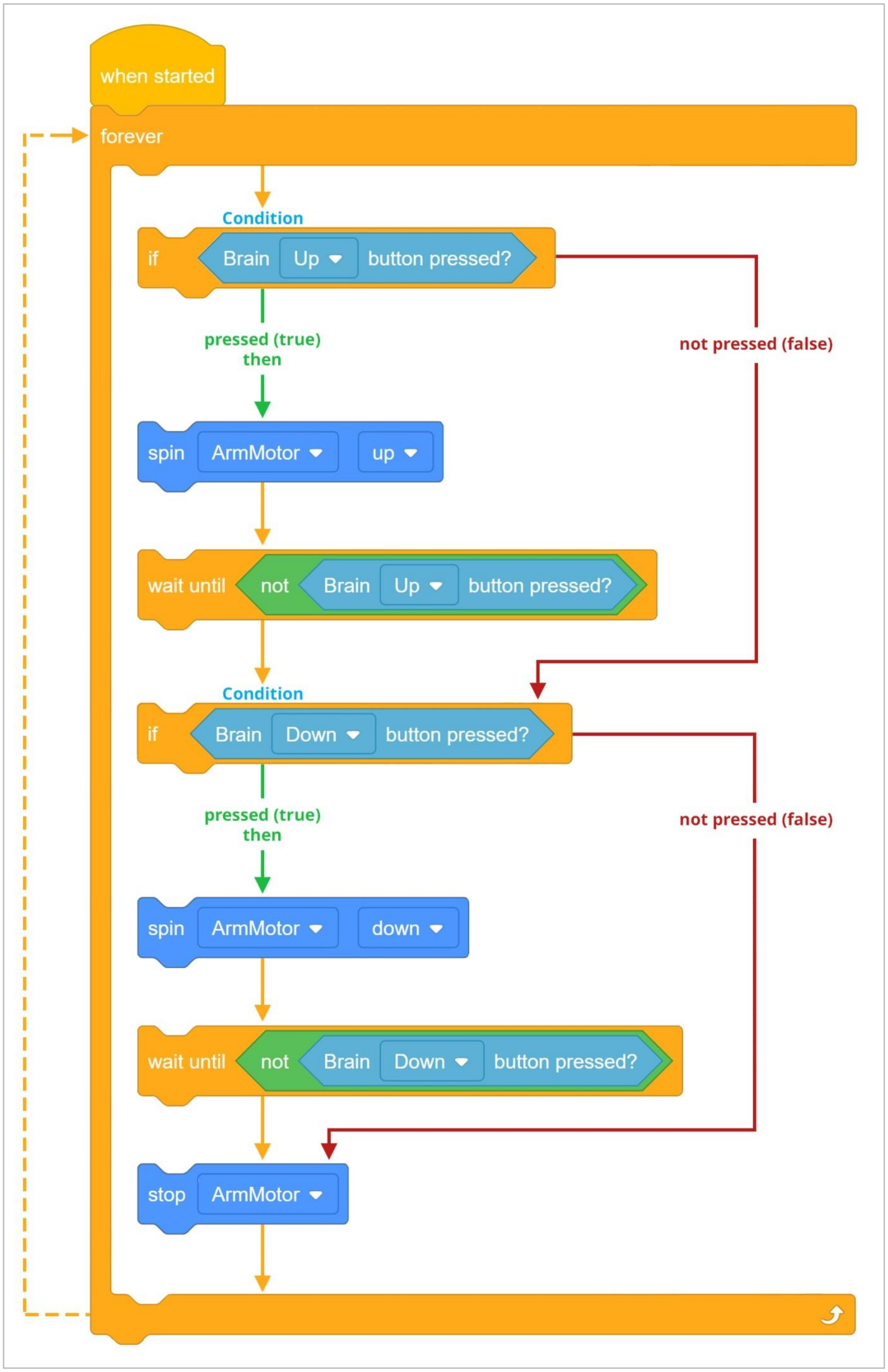
```
when started
forever
    if <Brain Up button pressed?> then
        spin ArmMotor up
        wait until <not <Brain Up button pressed?>>
    if <Brain Down button pressed?> then
        spin ArmMotor down
        wait until <not <Brain Down button pressed?>>
    stop ArmMotor
```

- Save the project as ArmUpDown. For help, watch the **Naming and Saving Your Project** tutorial video (Windows, Mac, iPad, Chrome) in VEXcode IQ Blocks.

- Check to make sure the project name **ArmUpDown** is now in the window in the center of the toolbar.
- Predict what the project will have the Clawbot do in your engineering notebook. Explain both the user's and the Clawbot's behaviors.
- Test to see if your prediction of what the project has the Clawbot do is correct.
    - Download the project to Slot 3 on the Clawbot, and then run it.
    - For help downloading and running a project, see the tutorial in VEXcode IQ Blocks that explains how to Download and Run a Project.
- Check your explanations of the project and add notes to correct them as needed.

2. Understanding the project flow.

In the previous step, the project did raise and lower the arm. However, since the condition of the Brain Up button begin pressed is first, if the Brain Up button is held down and then the Brain Down is also pressed, the arm will continue to spin up since that action will not stop until the Brain Up button is released. View the following flowchart that explains the project flow.

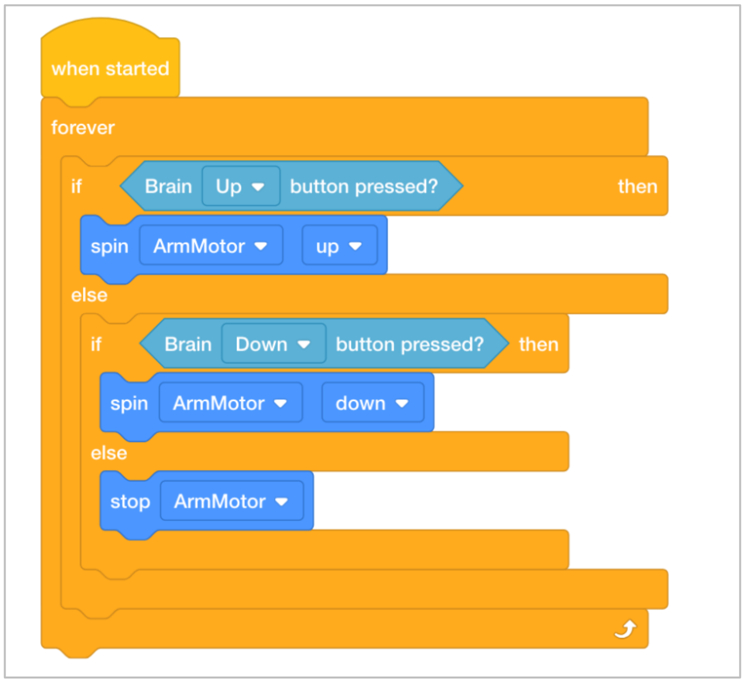

when started
forever
Condition
if
Brain
Up
button pressed?
pressed (true)
then
not pressed (false)
spin
ArmMotor
up
wait until
not
Brain
Up
button pressed?
Condition
if
Brain
Down
button pressed?
pressed (true)
then
not pressed (false)
spin
ArmMotor
down
wait until
not
Brain
Down
button pressed?
stop
ArmMotor

Thus, the project can replace the *if then* blocks with *if then else* blocks so that only one instance can be true at any time.

View the following tutorial video on *if then else* blocks:

Using *if then else* blocks will also get rid of the need for the *wait until* blocks because the arm will continue to spin up until the Brain Up button is released. This occurs because the "else" part of the *if then else* is never reached until the Brain Up button condition is false (released).

The *forever* block allows the first condition to be continuously checked.

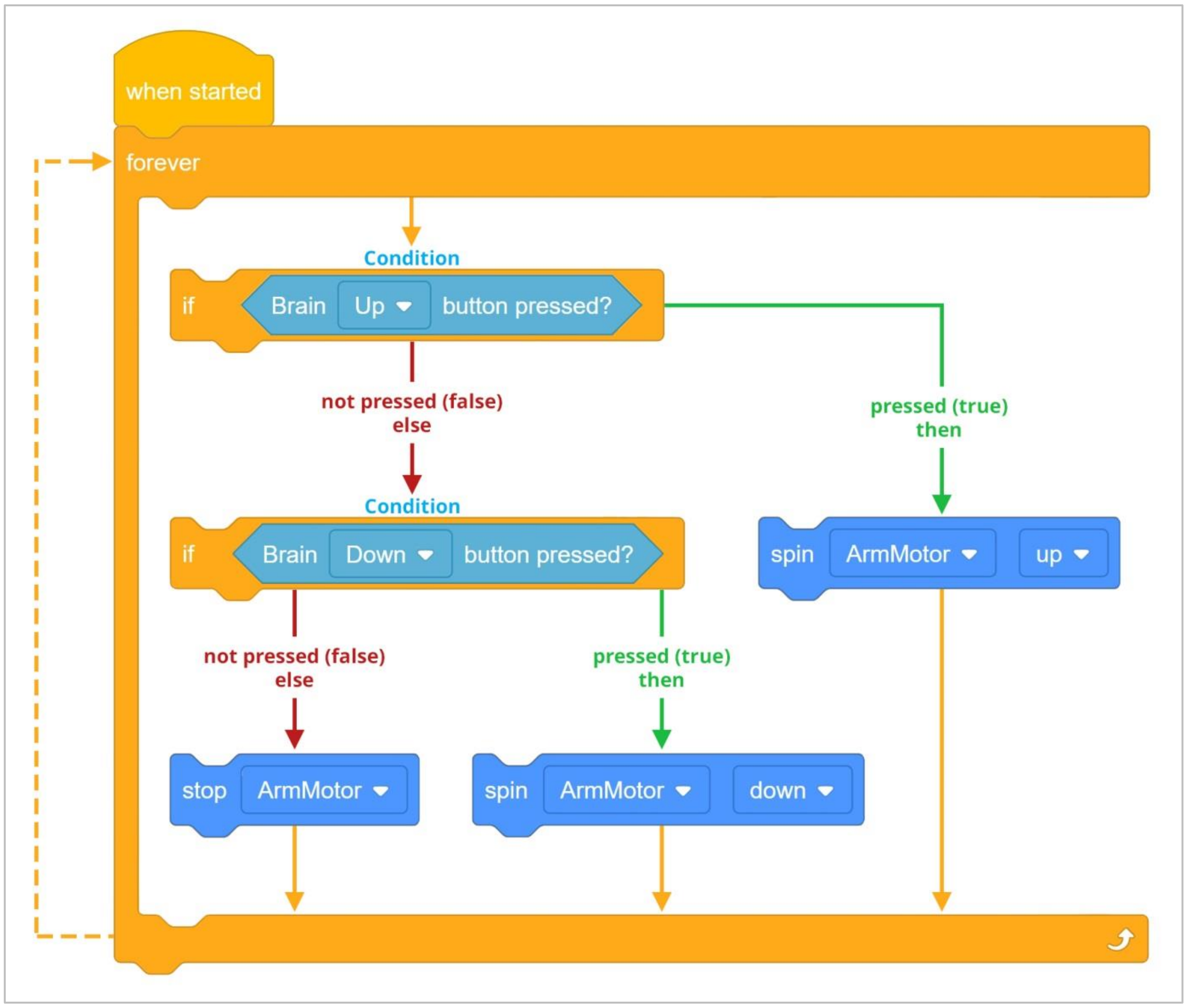


```
when started
forever
    if Brain Up button pressed? then
        spin ArmMotor up
    else
        if Brain Down button pressed? then
            spin ArmMotor down
        else
            stop ArmMotor
```

Let's explore changing the project by adding *if then else* blocks. The Arm Motor will now continue to spin until the Brain Up button is no longer pressed (released). Due to the *forever* block, this condition will continue to be checked until it is false.

Once the condition of the Brain Up button is false, the project will continue to the next block, which is to check the condition if the Brain Down button is pressed. If the condition of the Brain Down button being pressed is true, then the arm will spin up. Again, due to the *forever* block, this condition will continue to be checked until it is false.

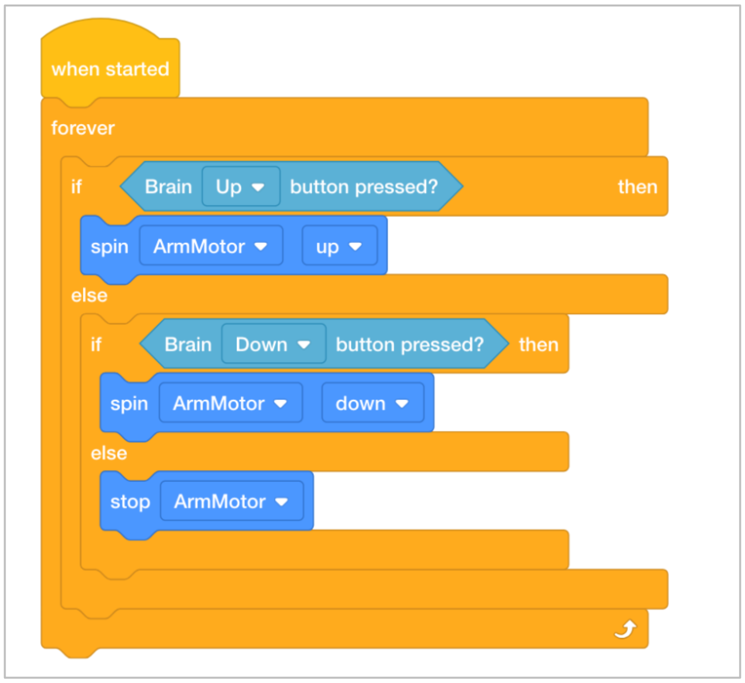
Once the condition of the Brain Up button is false, the project will continue to the next block, which is the *stop motor* block. Thus, the Arm Motor will only stop when both conditions are false (neither button is pressed).



3. Programming with *if then else* blocks.

Let's use the *if then else* blocks:

- Add the *if then else* blocks to your ArmUpDown project so that your project looks like the following:



```
when started
forever
    if <Brain Up button pressed?> then
        spin ArmMotor up
    else
        if <Brain Down button pressed?> then
            spin ArmMotor down
        else
            stop ArmMotor
```

- Save the project as ArmUpDown2. For help, watch the **Naming and Saving Your Project** tutorial video (Windows, Mac, iPad, Chrome) in VEXcode IQ Blocks.

- Download the project to Slot 4 on the Clawbot, and then run it.
- For help downloading and running a project, see the tutorial in VEXcode IQ Blocks that explains how to Download and Run a Project.
- Test to see if the arm will now spin up when the Brain Up button is pressed and spin down when the Brain Down button is pressed.
- Verify that when the Brain Up and Down buttons are not pressed (released) that the Arm Motor stops.
- Write your observations of how the Clawbot behaved before and after adding the if then else blocks to your project in your engineering notebook.

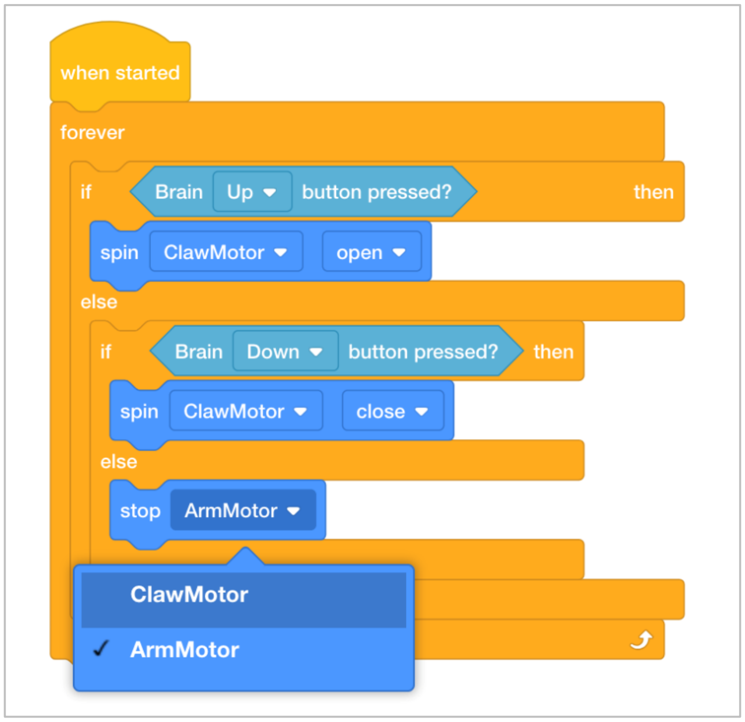
4. Programming the Claw.

In the previous step, the project was optimized to run more efficiently by using *if then else* blocks. In the previous example, the arm was manipulated using the Brain Up and Down buttons.

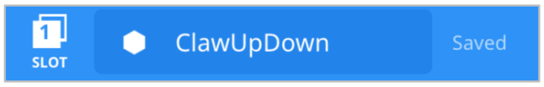
Using the exact same project outline, the Claw can also be manipulated.

- Review the ArmUpDown2 project and change the *spin* blocks to ClawMotor instead of ArmMotor by using the dropdown menu.

Ensure that the Brain Up button spins the Claw "open" and that the Brain Down button spins the Claw "close" by using the dropdown menus.


```
when started
forever
    if Brain Up button pressed? then
        spin ClawMotor open
    else
        if Brain Down button pressed? then
            spin ClawMotor close
        else
            stop ArmMotor
```

- Save the project as ClawUpDown. For help, watch the **Naming and Saving Your Project** tutorial video (Windows, Mac, iPad, Chrome) in VEXcode IQ Blocks.



- Download the project to Slot 1 on the Clawbot, and then run it.
- For help downloading and running a project, see the tutorial in VEXcode IQ Blocks that explains how to Download and Run a Project.
- Test to see if the Claw will now spin open when the Brain Up button is pressed and spin closed when the Brain Down button is pressed.
- Verify that when the Brain Up and Down buttons are not pressed (released) that the Claw Motor stops.
- Write your observations of how the Clawbot behaved before and after adding the if then else blocks to your project in your engineering notebook.

SPARK

APPLY

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

User Interfaces

Interacting with Computer Systems

The buttons you used on the brain are the beginning of a basic User Interface (UI). A UI is a space that allows the user to interact with a computer system (or machine). When you programmed the buttons on the brain, you gave users a way to interact with the Clawbot so they could raise and lower the arm. There are other types of User Interfaces (UIs), including Graphical User Interfaces (GUIs) such as touchscreens in cars and on smartphones. When you interact with a touchscreen on one of your devices (tablet, smartphone, smartwatch), those screens are often the only interface you have. Maybe your device has volume or power buttons as well but you mainly interact with the screen.

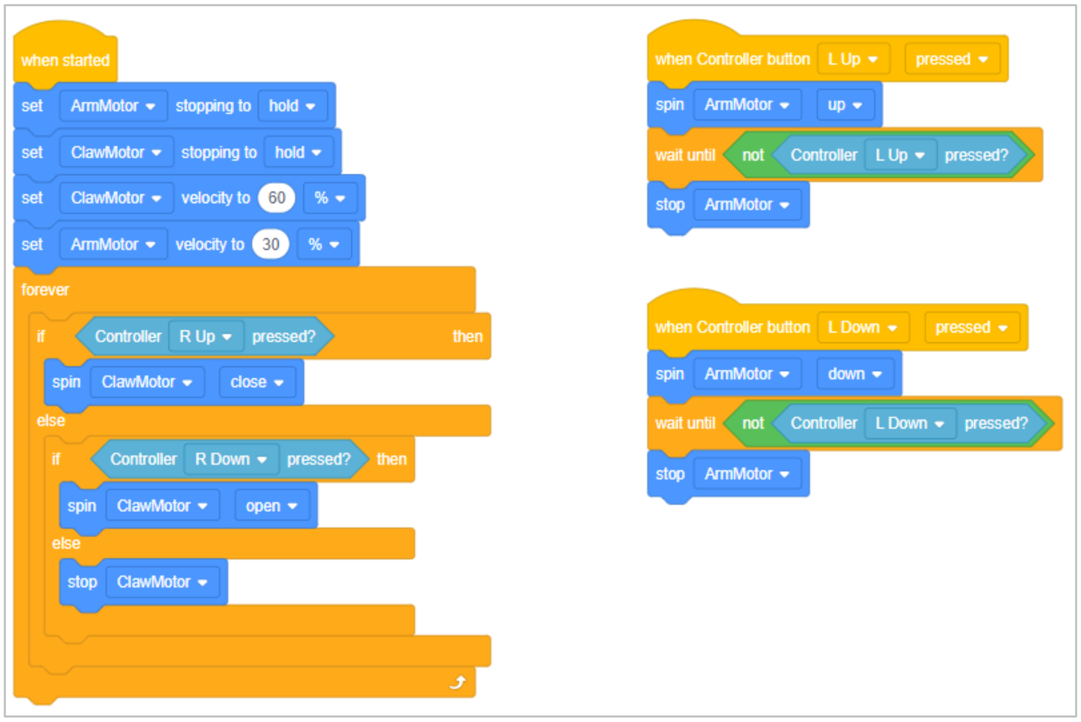
There are buttons on a TV remote that are programmed to turn the TV off or to turn the volume up when they are pushed. Some examples of UIs include the buttons on a video game controller or the buttons on a microwave. The ways these User Interfaces are designed depend on how the device works and how users interact with it.

Those design principles form the foundation of the User Experience (UX) while using a UI.

The User Experience is how well the interface lets me, as the user, do what I'm trying to do. Is the interface working as I expect it to? Is it responsive to what I'm trying to communicate with my presses? Is it organized well, or should future versions of the UI move the buttons around to make it easier? What does the interface look like in general? Is it pleasing to look at and does it make me want to use it more often?

When a UI is still being developed and undergoing iterations, the developers collect data on what works as planned and what needs to be fixed or enhanced. That data then informs the next round of iterative design. Some of the UX changes recommended occur before the release of the device. But, the device might also be sold as is and those changes are made later before the next version is offered to the public consumer.

The Controller as a User Interface



```
when started
set ArmMotor stopping to hold
set ClawMotor stopping to hold
set ClawMotor velocity to 60 %
set ArmMotor velocity to 30 %
forever
    if Controller R Up pressed? then
        spin ClawMotor close
    else
        if Controller R Down pressed? then
            spin ClawMotor open
        else
            stop ClawMotor

when Controller button L Up pressed
spin ArmMotor up
wait until not Controller L Up pressed?
stop ArmMotor

when Controller button L Down pressed
spin ArmMotor down
wait until not Controller L Down pressed?
stop ArmMotor
```

The Controller Buttons example project from VEXcode IQ Blocks

Remotely Controlling the Robot

We most often use remote controls to interact with our televisions. We press buttons that make the television display a channel or information/access screen that we want. Technically, your television's remote control is a UI. However, it is a much less sophisticated UI than the one that your smartphone uses.

Programming your IQ Controller is much more sophisticated. During the Driver Controlled matches of a competition, you want your driver/team to have as many advantages as possible. You can program the buttons and joysticks to do more than one simple behavior, and you can program them to do complex behaviors when buttons/joysticks are used in combination - similar to how some gaming controllers work. As the programmer of your Controller, you consider which buttons to use in combination by figuring out how your fingers and hands would need to be placed in order to reach all of the buttons involved.

The image above shows the project from the Controller Buttons example project from VEXcode IQ Blocks. Do you see how the *forever* loop has two *if then else* blocks nested inside of it? It makes the project check whether the R Up or R Down buttons are pressed, and the robot spins the Claw Motor closed or open based on one being pressed. That *forever* loop is very important. It makes your IQ Controller check repeatedly on which button(s) is being pressed so that the robot performs the appropriate behavior(s).

Consider that you could put two *pressing controller* Sensing blocks in an *and* Operators block. That would have the project check if two buttons were being pressed. You could also put an *and* block inside of another *and* block and have three conditions being checked before a behavior is carried out. This would let you program sequences of behavior that can be started simply by pressing buttons on your controller.

Consider how many more combinations of conditionals you could create with all of the buttons and their combinations on the Controller. Of course, as you program more complex behaviors into the functioning of the Controller, the project gets closer to having the robot be autonomous. A competition team needs to figure out which are the best behaviors to program into their Controllers as complex sequences and which behaviors are best left decomposed into multiple parts so that the Controller lets the driver (user) have more control over the speed and accuracy of the behavior.

SPARK
RETHINK

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 User Interface Challenge

An Interface to Grab and Lift!

In the User Interface Challenge, you need to program your robot so that a user can use the Check, Up, and Down buttons on the brain to pick up a variety of objects.

Your user interface for controlling the Clawbot will need:

- A button or buttons for opening the claw
- A button or buttons for closing the claw
- A button or buttons for raising the arm
- A button or buttons for lowering the arm

To complete the challenge you will need:

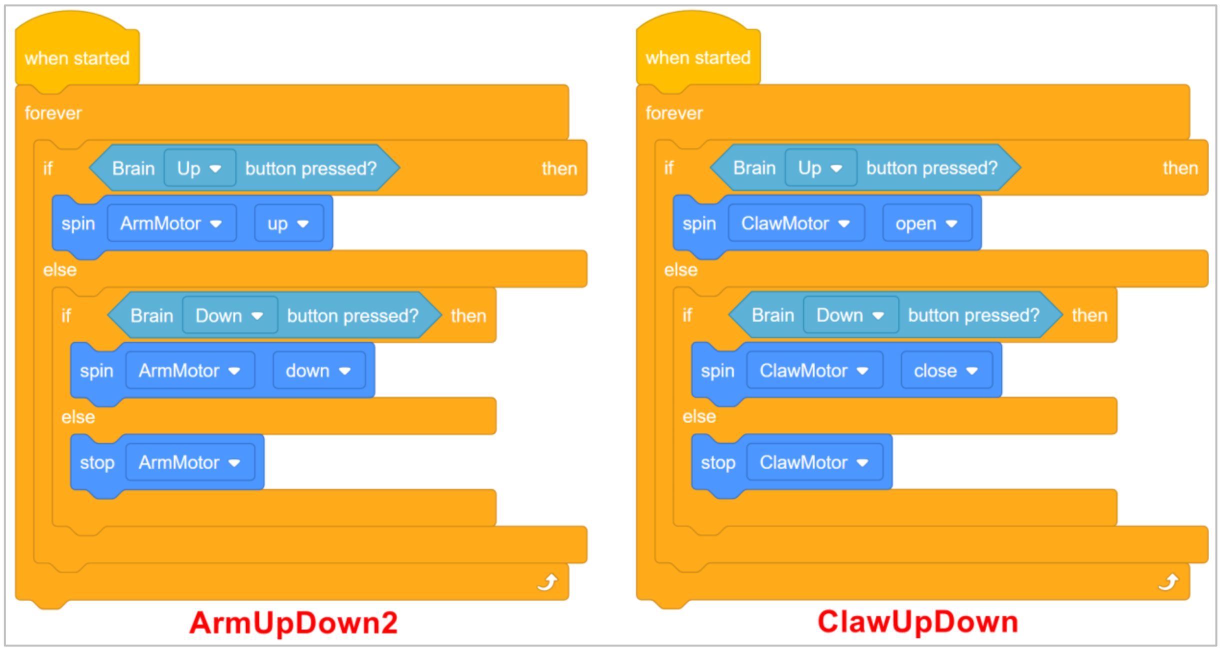
- A Clawbot

- Objects to be picked up: an empty can or water bottle, a VEX cube, an unused piece from the VEX kit, or anything else that your teacher can provide

Design, Develop, and Iterate on your Project

In this section, you will use the projects that you build in the Play section to create a project that can manipulate both the arm and the claw.

Recall the ArmUpDown2 and the ClawUpDown projects.



```
when started
forever
    if <Brain Up button pressed?> then
        spin ArmMotor up
    else
        if <Brain Down button pressed?> then
            spin ArmMotor down
        else
            stop ArmMotor
```

ArmUpDown2

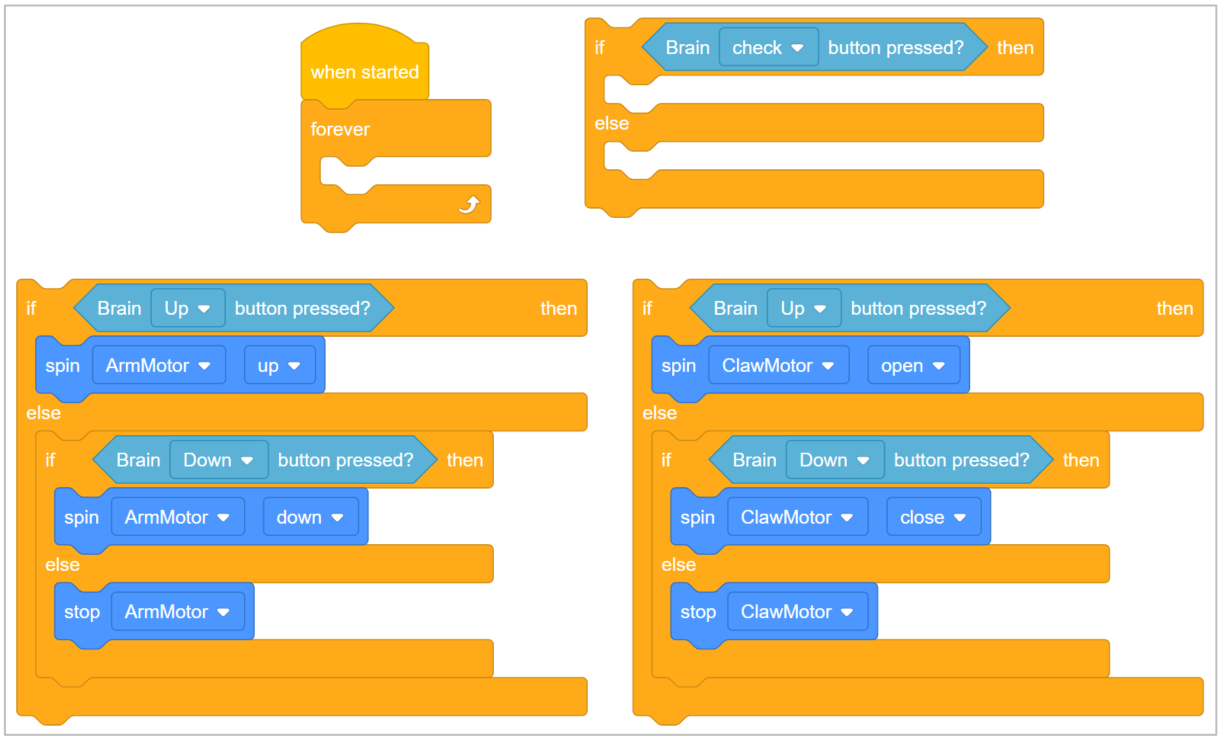
```
when started
forever
    if <Brain Up button pressed?> then
        spin ClawMotor open
    else
        if <Brain Down button pressed?> then
            spin ClawMotor close
        else
            stop ClawMotor
```

ClawUpDown

We want to somehow incorporate both of these projects into the same project. However, there is only one Brain Up button and one Brain Down button.

Thus, we need a button to act as a "switcher" between the arm and claw.

Use the following outline of blocks to help you build your project:



```
when started
forever

if <Brain check button pressed?> then
else

if <Brain Up button pressed?> then
    spin ArmMotor up
else
    if <Brain Down button pressed?> then
        spin ArmMotor down
    else
        stop ArmMotor

if <Brain Up button pressed?> then
    spin ClawMotor open
else
    if <Brain Down button pressed?> then
        spin ClawMotor close
    else
        stop ClawMotor
```

Answer the following questions in your engineering notebook as you plan your project:

- What do you want to program the robot to do? Explain with details.
- How many conditions will your project need to check in the *if then else* blocks?

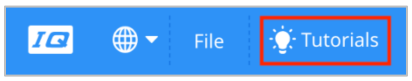
  Hint: Use the Brain Check button as the "switcher" between the arm and the claw. Thus, if the Brain Check button is pressed and held, the arm is controlled using the Brain Up and Down buttons.
  If the Brain Check Button is released, the claw is controlled using the Brain Up and Down buttons.

Follow the steps below as you create your project:

- Plan out the conditions that your project needs to check using drawings and pseudocode.
- Use the pseudocode you created to develop your project.
- Test your project often and iterate on it using what you learned from your testing.
- What could you add to your project to better control the Claw and Arm Motors? Explain with details.
- Share your final project with your teacher.

If you're having trouble getting started, review the following in VEXcode IQ Blocks:

- If Then Else Blocks or Using Loops tutorial videos
- Using Help tutorial video
- Previous versions of your project (**ArmUpDown2** or **ClawUpDown**)


File
Tutorials

The User Interface Challenge

The User Interface Challenge

In the User Interface Challenge, you will program the Clawbot so that a user can press the brain's Check, Up, and Down buttons to control the arm and claw motors. Then those buttons will be used to pick up and replace a variety of ten objects. This challenge does not require the Clawbot to drive or turn. The objects are picked up and then replaced to the same spot on the table or floor.

Rules:

- One button (Up button) or one combination of buttons (Check button and Up button) must only do one of the four actions: open the claw, close the claw, lift the arm, or lower the arm.
- Using the Controller is not allowed.
- Each Clawbot will need to lift and replace as many objects as possible within one minute and without dropping them. Lifting and replacing one object at a time is recommended.
  - The one-minute round ends at the 1-minute mark or if any object is dropped - even if the round is only a few seconds in. Dropping an object disqualifies the team from the full minute of the round but any points earned prior to the drop are counted.
- If all of the provided objects have been lifted before the one-minute round is over, objects can be re-used until time is called.

- The object needs to be lifted higher than the arm's motor before it is replaced on the table.
- Each object successfully grabbed and lifted up then down and replaced is worth one point.
- Between rounds, roles can be changed but only one user per Clawbot can play each round.
- The group with the most points at the end of all of the rounds, wins!

Round Number	Clawbot ID	Number of Items Successfully Lifted and Replaced

Roles in the Challenge:

- There should only be one user (the ***lifter***) lifting the objects during each round but groups can switch users between rounds.
- There should be a designated ***scorekeeper*** who tallies how many objects each Clawbot is able to lift successfully. Each object successfully replaced is worth one point. After an object is dropped, the Clawbot's turn is over. Multiple rounds can be played. A scoring table is included here but rows can be added for additional groups and rounds. Each group can use their own scoring table or everyone can record on the same one. If multiple groups are using the same score table, write each groups' Clawbot ID or group name in the second column. Also, the scorekeeper should combine the points from each round at the end of the User Interface Challenge for a total score.

- There should be a person assigned to switch objects out from the table: the ***switcher***. Because the Clawbot is not turning or driving, the lifted object will need to be removed and replaced with a new object after each attempt.
- There should be a person assigned to keep track of the time: the ***timekeeper***. Each round is one minute.
- The teacher should provide the objects approved for this challenge prior to starting.

SPARK

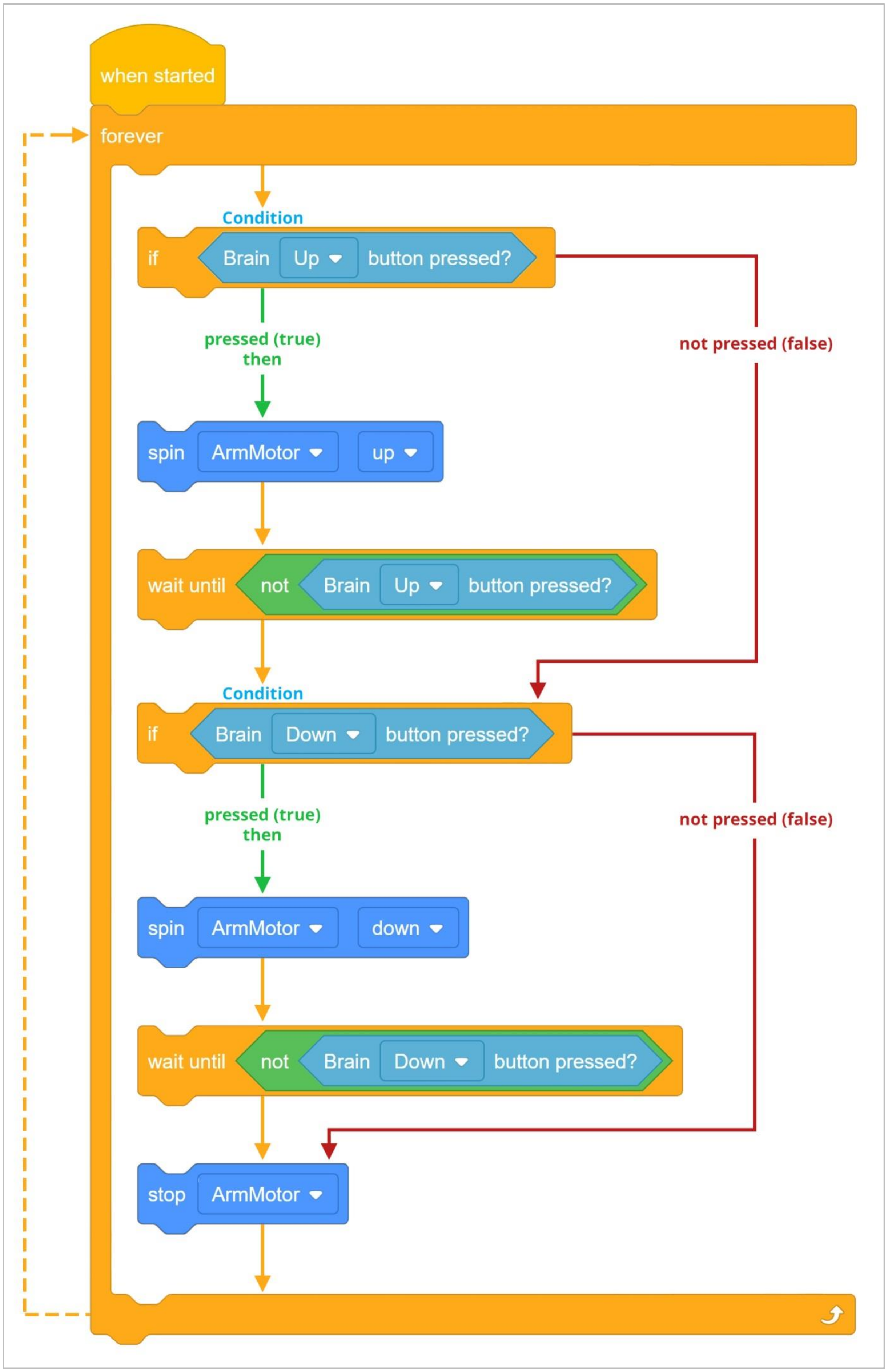
KNOW

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

Review

1. **Why would a programmer use conditionals within a project?**
    - ○ To have the robot loop behaviors
    - ○ To have the robot spin its motors in a sequence
    - ○ To have the robot make decisions
    - ○ To have the robot play sounds

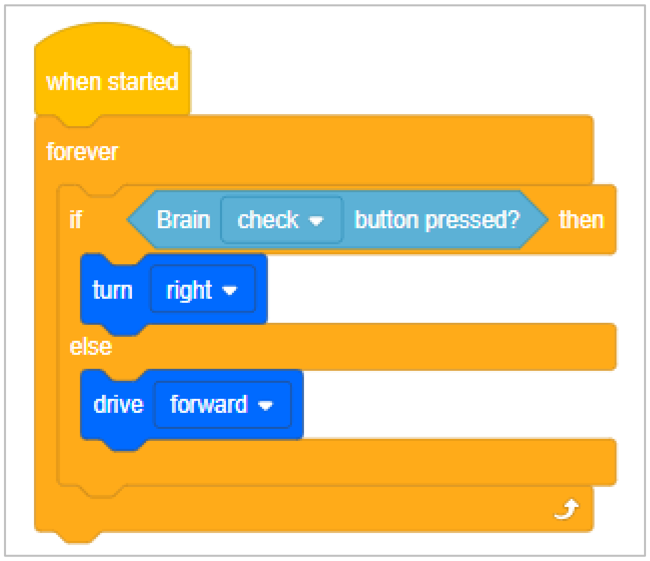
2. **In this project, according to the project flow, what is the next block that the project will run if the first condition of the Brain Up button being pressed is false (released)?**



when started
forever
Condition
if
Brain
Up
button pressed?
pressed (true)
then
not pressed (false)
spin
ArmMotor
up
wait until
not
Brain
Up
button pressed?
Condition
if
Brain
Down
button pressed?
pressed (true)
then
not pressed (false)
spin
ArmMotor
down
wait until
not
Brain
Down
button pressed?
stop
ArmMotor

- The second condition of the Brain Down button being pressed will be checked.
- The *spin* block.
- The *wait until* block.
- The *stop motor* block.

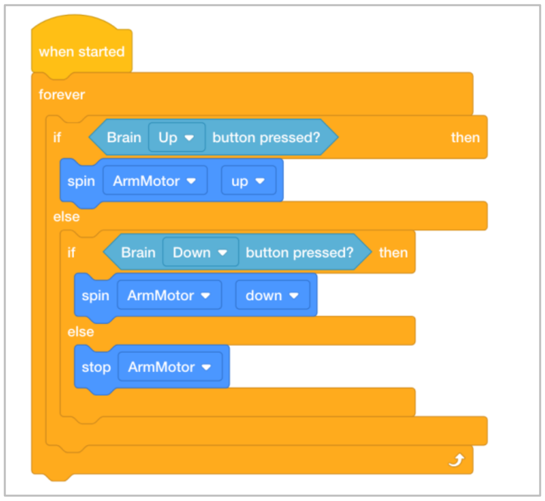
3. This project is running and the robot turned right. What condition is TRUE?



```
when started
forever
    if < Brain check ▾ button pressed? > then
        turn right ▾
    else
        drive forward ▾
```

- The Check button is pressed.
- The Up button is pressed.
- The Check button is NOT pressed.
- The Up button is NOT pressed.

4. Which of the following is the best pseudocode for this project?



```
when started
forever
    if < Brain Up ▾ button pressed? > then
        spin ArmMotor ▾ up ▾
    else
        if < Brain Down ▾ button pressed? > then
            spin ArmMotor ▾ down ▾
        else
            stop ArmMotor ▾
```

- If the Brain Up button is pressed (true) and the Brain Down is pressed (true), lower the arm.
  If the Brain Up button is pressed (true) and Down button is released (false), raise the arm.
  If the Brain Up or Down buttons are pressed (true), stop the Arm Motor.
- Raise the arm when two buttons are pressed.
  Lower the arm when one button is pressed.
  Turn right when no buttons are pressed.
- If the Brain Up button is pressed (true), then spin the arm up. If it is false (released) then check if the Brain Down button is pressed.
  If the Brain Down button is pressed (true), spin the arm down. If it is false (released) then stop the arm.
- Use the Check, Up, and Down buttons as a UI.

5. Which of the following is NOT a type of User Interface (UI)?

- IQ Controller
- A doorknob
- Smartphone touchscreen
- Buttons on a microwave

APPENDIX

Additional information, resources, and materials.

Knowledge Base Articles

Links to the VEX Robotics Knowledge Base Articles for this STEM Lab:

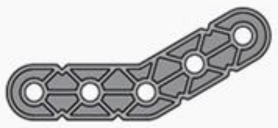
- How to Turn On/Off a VEX IQ Robot Brain
  https://kb.vex.com/hc/en-us/articles/360035952571-How-to-Turn-On-Off-a-VEX-IQ-Robot-Brain
- How to Read Indicator Lights on the VEX IQ Robot Brain
  https://kb.vex.com/hc/en-us/articles/360035590672-How-to-Read-Indicator-Lights-on-the-VEX-IQ-Robot-Brain
- How to Navigate the VEX IQ Robot Brain
  https://kb.vex.com/hc/en-us/articles/360035952331-How-to-Navigate-the-VEX-IQ-Robot-Brain
- How to Connect VEX IQ Devices to Smart Ports
  https://kb.vex.com/hc/en-us/articles/360035952151-How-to-Connect-VEX-IQ-Devices-to-Smart-Ports
- How to Install or Remove the VEX IQ Robot Battery
  https://kb.vex.com/hc/en-us/articles/360035951991-How-to-Install-or-Remove-the-VEX-IQ-Robot-Battery
- How to Charge the VEX IQ Robot Battery
  https://kb.vex.com/hc/en-us/articles/360035955011-How-to-Charge-the-VEX-IQ-Robot-Battery
- How to Use the Autopilot Program in the Demos Folder
  https://kb.vex.com/hc/en-us/articles/360035952031-How-to-Use-the-Autopilot-Program-in-the-Demos-Folder
- Best Practices for Preserving the VEX IQ Robot Battery’s Life
  https://kb.vex.com/hc/en-us/articles/360035953671-Best-Practices-for-Preserving-the-VEX-IQ-Robot-Battery-s-Life
- Ideas for Organizing the VEX IQ Super Kit
  https://kb.vex.com/hc/en-us/articles/360035590332-Ideas-for-Organizing-the-VEX-IQ-Super-Kit
- VEX IQ Brain Status (USB Cable)
  https://kb.vex.com/hc/en-us/articles/360035955411-How-to-Understand-the-VEX-IQ-Brain-Status-Icon-USB-VEXcode-IQ-Blocks

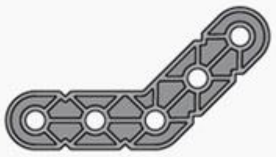
Links to VEXCode IQ Blocks Knowledge Base Articles for this STEM Lab:

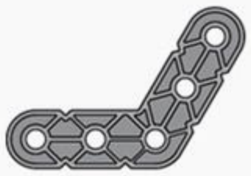
- How to Begin a New Project in VEXcode IQ Blocks
  https://kb.vex.com/hc/en-us/articles/360035954551-How-to-Begin-a-New-Project-VEXcode-IQ-Blocks

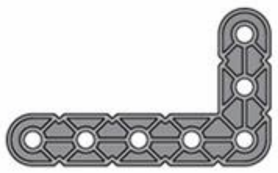
- How to Download and Run a Project
  https://kb.vex.com/hc/en-us/articles/360035591232-How-to-Download-and-Run-a-Project-VEXcode-IQ-Blocks
- How to Save a Project on Windows
  https://kb.vex.com/hc/en-us/articles/360035954531-How-to-Save-a-Project-on-Windows-VEXcode-IQ-Blocks
- How to Save a Project on macOS
  https://kb.vex.com/hc/en-us/articles/360035954511-How-to-Save-a-Project-on-macOS-VEXcode-IQ-Blocks
- How to Save a Project on Chromebook
  https://kb.vex.com/hc/en-us/articles/360035955351-How-to-Save-on-a-Chromebook-VEXcode-IQ-Blocks
- How to Download to a Selected Slot on the Brain
  https://kb.vex.com/hc/en-us/articles/360035591292-How-to-Download-to-a-Selected-Slot-on-the-Brain-VEXcode-IQ-Blocks

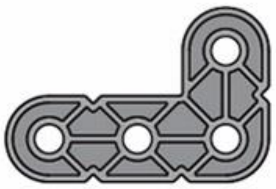
Identifying Angle Beams

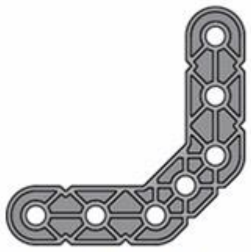

30 Degree Angle Beam
228-2500-147


45 Degree Angle Beam
228-2500-148


60 Degree Angle Beam
228-2500-149


3x5 Right Angle Beam
228-2500-150

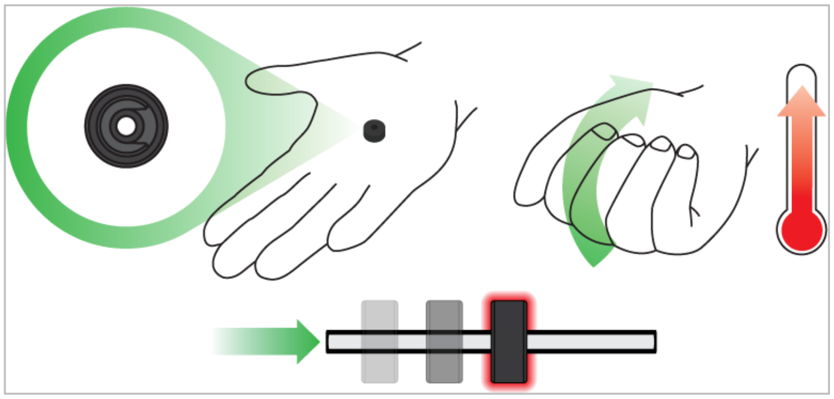

2x3 Right Angle Beam
228-2500-145


4x4 Offset Right Angle Beam 228-2500-151

How to Identify the Different Angles of the Angled Beams

There are four different types of beams that have a bend at an angle: 30^o Angle Beams, 45^o Angle Beams, 60^o Angle Beams, and Right Angle (90^o) Beams. There are also three types of Right Angle Beams: 3x5, 2x3, and Offset. The best way to tell which angles are which is to stack the beams on top of each other. Then you can compare how they look. You can also use a protractor to measure the angle of the beam.

Installing Rubber Shaft Collars

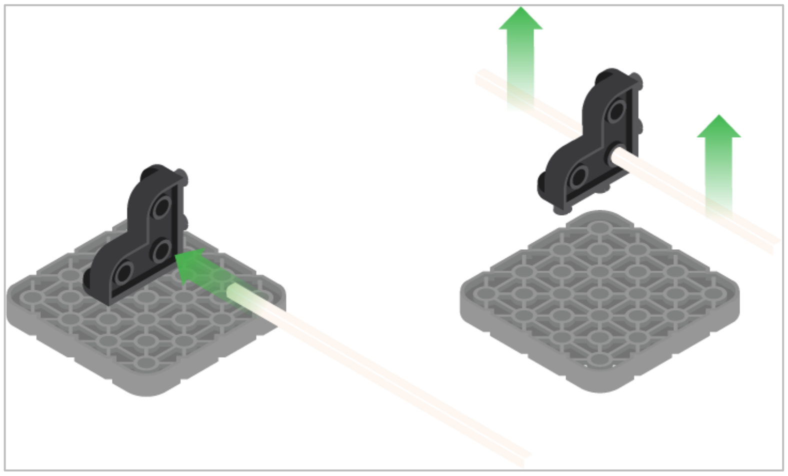


Using your hand to warm a Rubber Shaft Collar

Rubber Softens as it gets Warm

Hold the Rubber Shaft Collars in your hand for 15-30 seconds before you slide them onto a shaft. Holding the Rubber Shaft Collar in your hand will warm and soften the rubber to make it easier to slide onto a shaft.

Removing Connectors from Beams and Plates

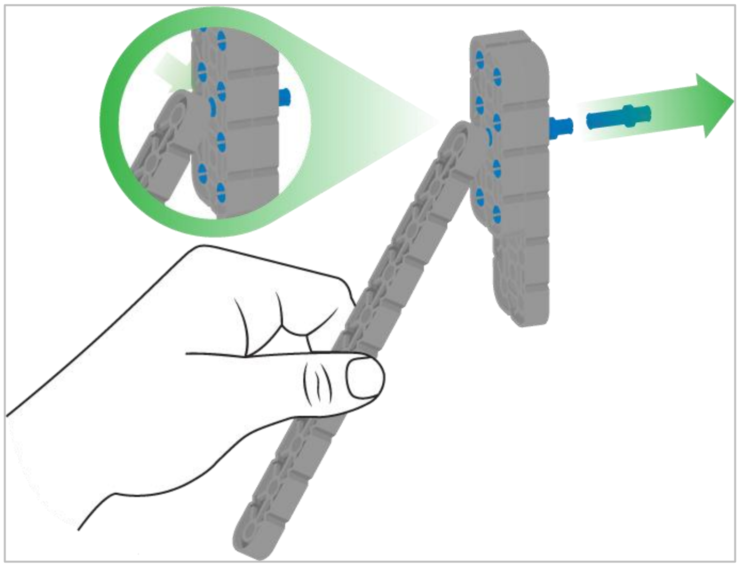


Using a pitch shaft to remove a corner connector

How to Easily Remove Connectors

You can easily remove corner connectors from beams or plates by placing a metal shaft through one of the holes of the corner connector and pulling outward while holding down the beam or plate.

Removing Pins from VEX IQ Beams and Plates

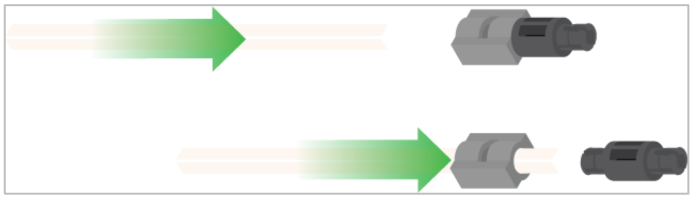


Removing a pin from a plate assembly using a beam

How to Easily Remove Pins from Beams and Plates

You can quickly remove connector pins from beams or plates by pressing a beam against the back of the pin, which partially pushes the pin out, so you can remove it with your fingers. You can use this technique to more easily remove pins from individual plates and beams, or from built structures.

Removing Standoffs from Mini Standoff Connectors

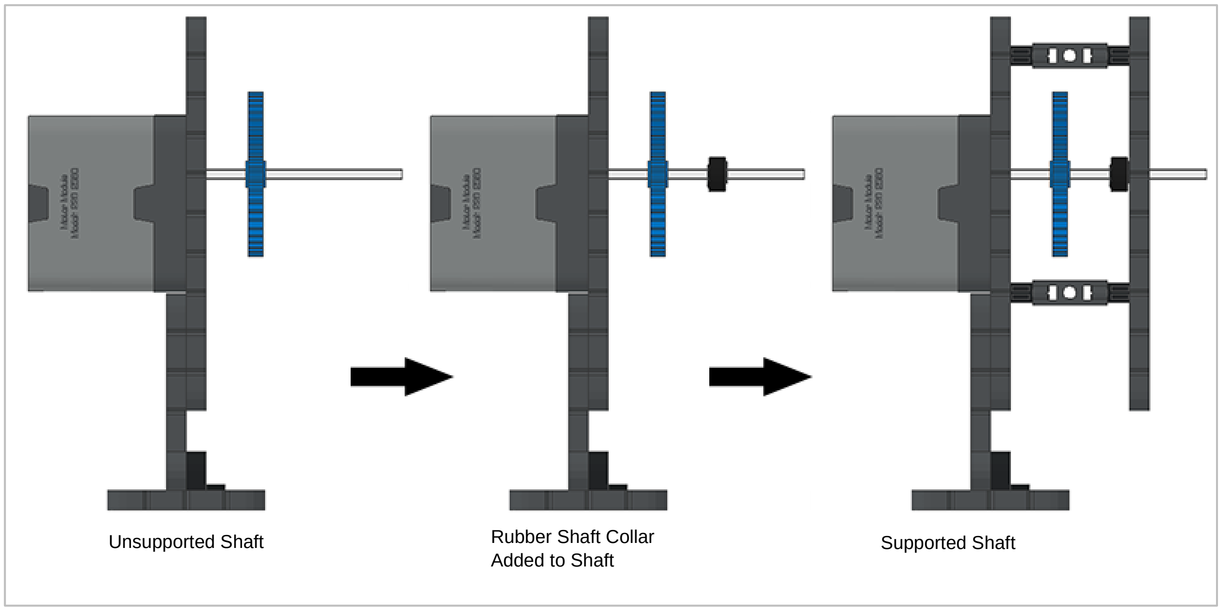


Removal of a standoff from a Mini Standoff Connector

How to Easily Remove Parts from Mini Standoff Connectors

Standoffs and Mini Standoff Connectors can be separated by pushing a shaft through the Mini Standoff Connector. The same technique can be used for parts with similar ends in Mini Standoff Connectors, such as pins.

Supporting Shafts using Rubber Shaft Collars

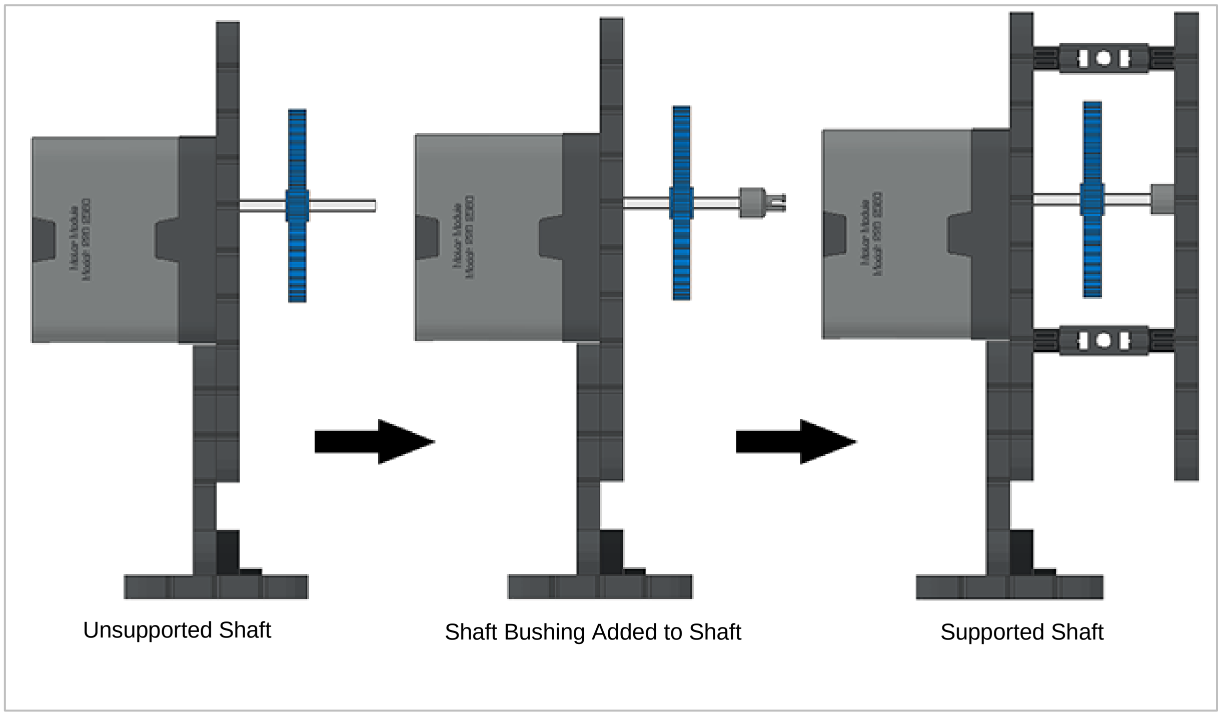


Supporting a shaft with a Rubber Shaft Collar

How to Support Shafts with Rubber Shaft Collars

Shafts can fall out of place or alignment very easily if they aren't supported properly. You can make a shaft more secure and prevent it from falling out of place by putting a Rubber Shaft Collar before the end of it. You can then connect the shaft to a support structure with the shaft collar positioned against it. That will allow the shaft to turn but will prevent it from wobbling or falling out.

Supporting Shafts using Shaft Bushings



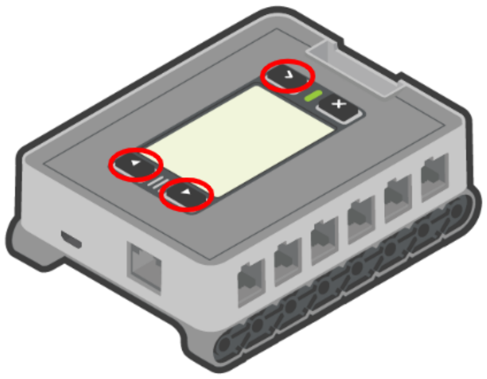
Supporting a shaft with a Shaft Bushing

How to Support Shafts Using Shaft Bushings

Shafts can fall out of place or alignment very easily if they aren't supported properly. You can make a shaft more secure and prevent it from falling out of place by putting a bushing at the end of it. You can then connect that bushing into another beam or additional part. That will allow the shaft to turn but will prevent it from wobbling or falling out.

Popups

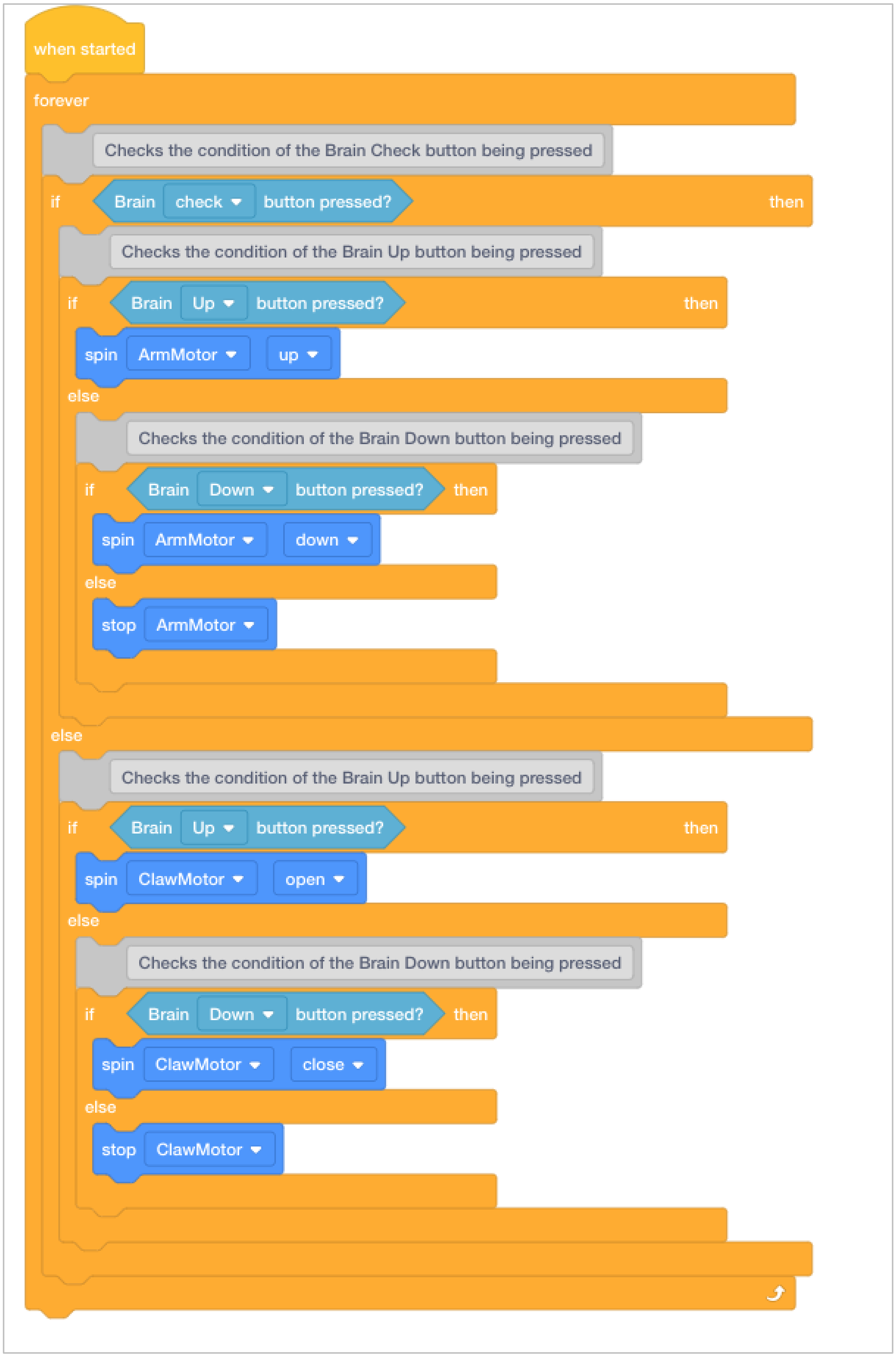
Pseudocode for the User Interface Challenge



Config: Clawbot with drivetrain

```
Loop forever
    If the Check button is pressed, then
        If the Up button is pressed, then
            Raise the arm
        Else
            If the Down button is pressed, then
                Lower the arm
            Else
                Stop the Arm Motor
    Else
        If the Up button is pressed, then
            Open the claw
        Else
            If the Down button is pressed, then
                Close the claw
            Else
                Stop the Claw Motor
```

Example Solution to User Interface Challenge



```
when started
forever
    Checks the condition of the Brain Check button being pressed
    if Brain check ▾ button pressed? then
        Checks the condition of the Brain Up button being pressed
        if Brain Up ▾ button pressed? then
            spin ArmMotor ▾ up ▾
        else
            Checks the condition of the Brain Down button being pressed
            if Brain Down ▾ button pressed? then
                spin ArmMotor ▾ down ▾
            else
                stop ArmMotor ▾
    else
        Checks the condition of the Brain Up button being pressed
        if Brain Up ▾ button pressed? then
            spin ClawMotor ▾ open ▾
        else
            Checks the condition of the Brain Down button being pressed
            if Brain Down ▾ button pressed? then
                spin ClawMotor ▾ close ▾
            else
                stop ClawMotor ▾
```