Co.Lab Robot Activity 1: Building the Bot & Getting it Running

Table of contents

Activity at a glance

Overview	In this activity, you assemble the Co.Lab robot, connect the micro:bit, download sample code, and run your robot for the first time.
Prerequisites	None, though it takes a little bit of dexterity (or help from a patient adult) to assemble the chassis and connect the wires.
Materials & Tools	**Materials** ● Red Hat Co.Lab Robot Kit (red.ht/robot-kit) **Tools** ● Computer with web access, or phone or tablet loaded with the micro:bit app (https://microbit.org/get-started/user-guide/mobile/) ● Permanent marker **Nice-to-have (but not necessary) supplies** ● Electrical tape - not necessary but makes one step of the assembly easier ● Slip-joint pliers - not necessary, but useful to grip parts during assembly
Background & Reference	Good references: ● Introduction to the micro:bit: https://microbit.org/get-started/user-guide/overview/ and https://www.youtube.com/watch?v=oNLf6aFYVoU&feature=emb_logo ● About the moto:bit: https://learn.sparkfun.com/tutorials/microbot-kit-experiment-guide/about-the-motobit-board?_ga=2.43162597.1517527125.1586905967-1522773540.1573599070 More projects: ● https://microbit.org/projects/ - many, many ideas to try with your micro:bit

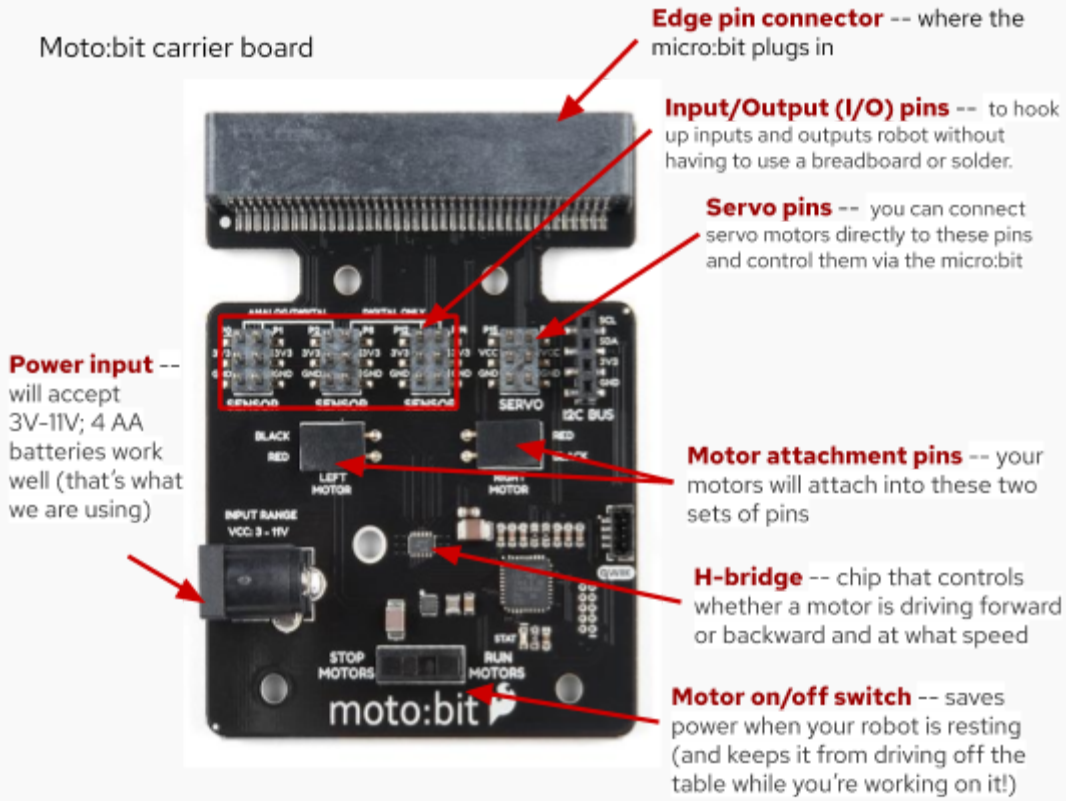
Directions

Identify the Parts & Read the Instructions

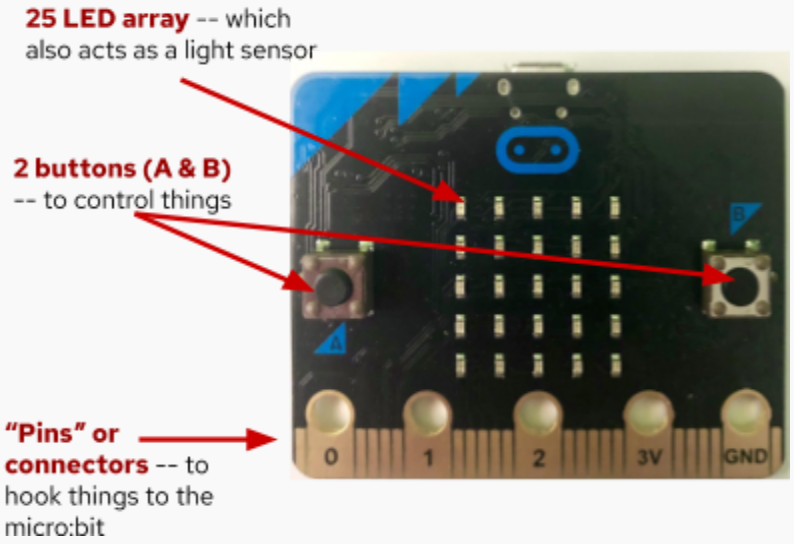
The first things I do when starting a project like this are to **make sure that I have all the pieces** (and that I know which pieces are which!) and **read through all the instructions** (before doing anything else).

So let's look at the robot parts.

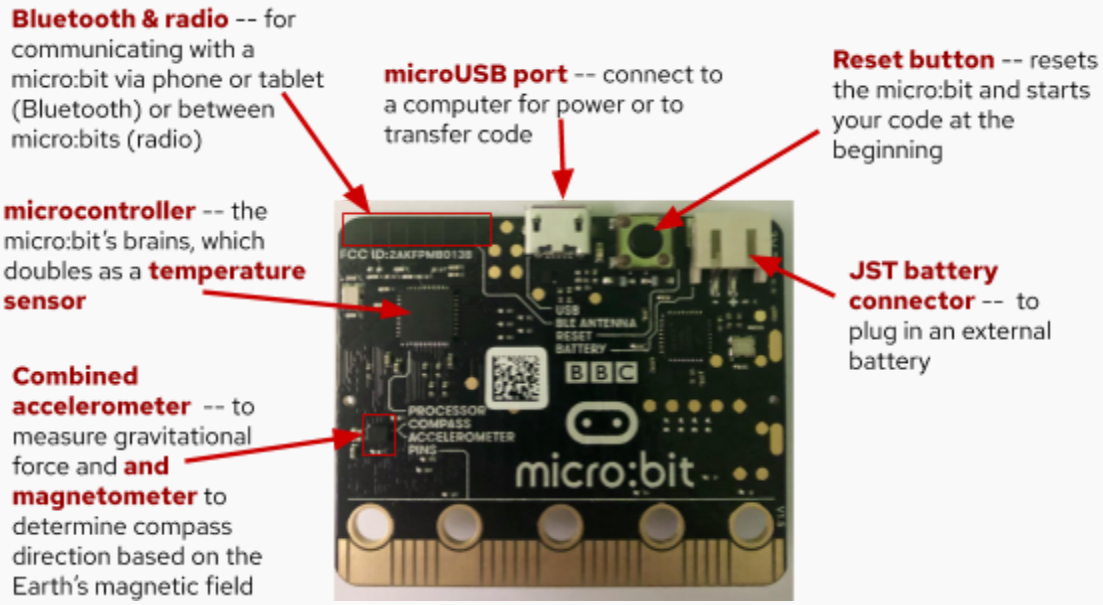
1. Find your moto:bit, the carrier board that the micro:bit plugs into. This picture identifies the parts of the moto:bit that will be important while building your robot.



2. Now, find your micro:bit -- the robot's "brains". This is the front:



And this is the back:



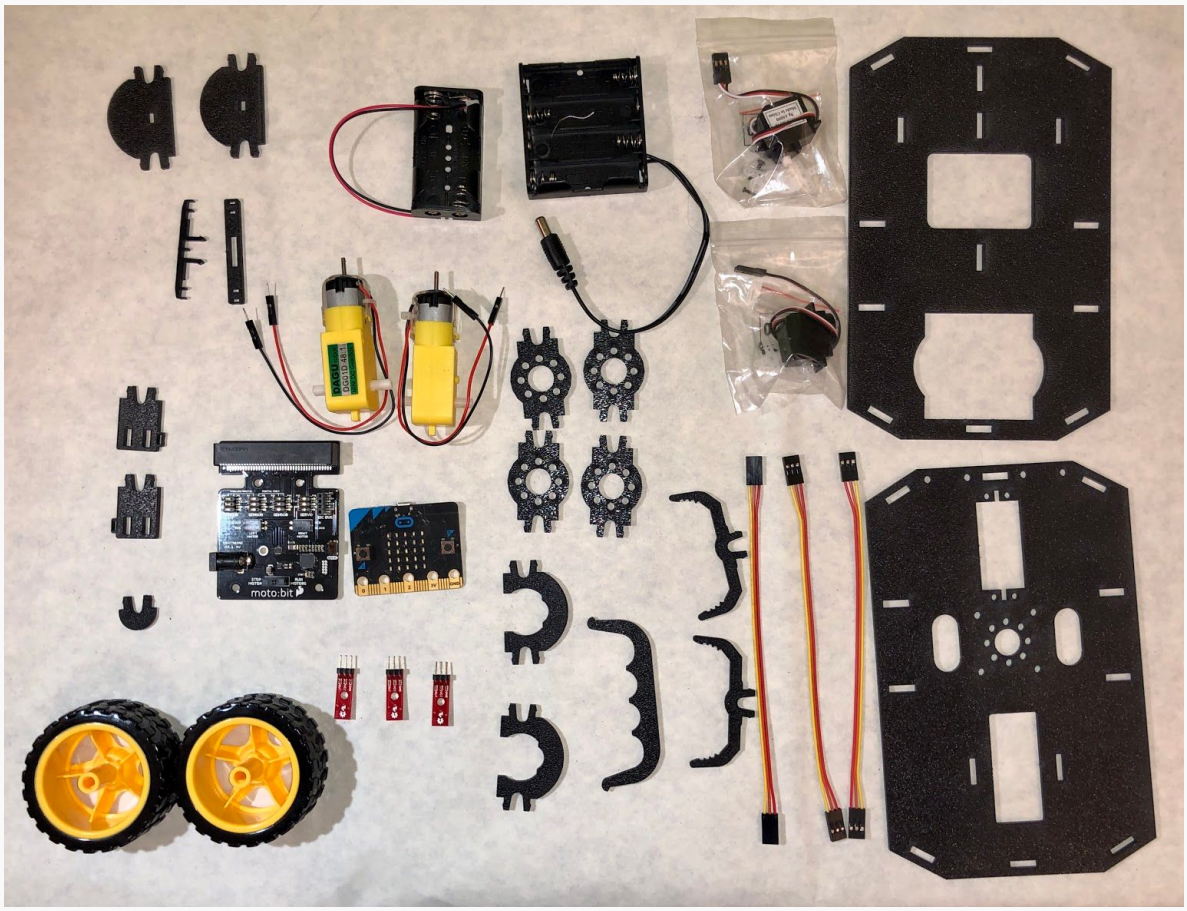
3. Next, we're going to need to snap out a few parts from the two main pieces of the chassis. Don't use pliers (or hammers or drills or ...). These should come out easily if you twist them a little in the frame as they are just held in by a thin piece. You should end up with seven new-and-shiny pieces, like these:

4. Finally, lay out all the parts on your worktable, placing all the parts of the same type together and counting them.

   Why do this?

   Well, unfortunately, sometimes instructions don't do a great job of specifying which of the somewhat-similar-but-slightly-different parts you should use on a given step (and sometimes there aren't pictures to help out). But if the instructions call for a certain number of that part, you can sometimes use the number of each part you received to help you figure out which part to use.

   Take a look at this picture, which should show almost the same pieces you now have -- you will also have some cool stickers that are brand new, a 6" microUSB cord, 4 AA batteries, and 2 AAA batteries):



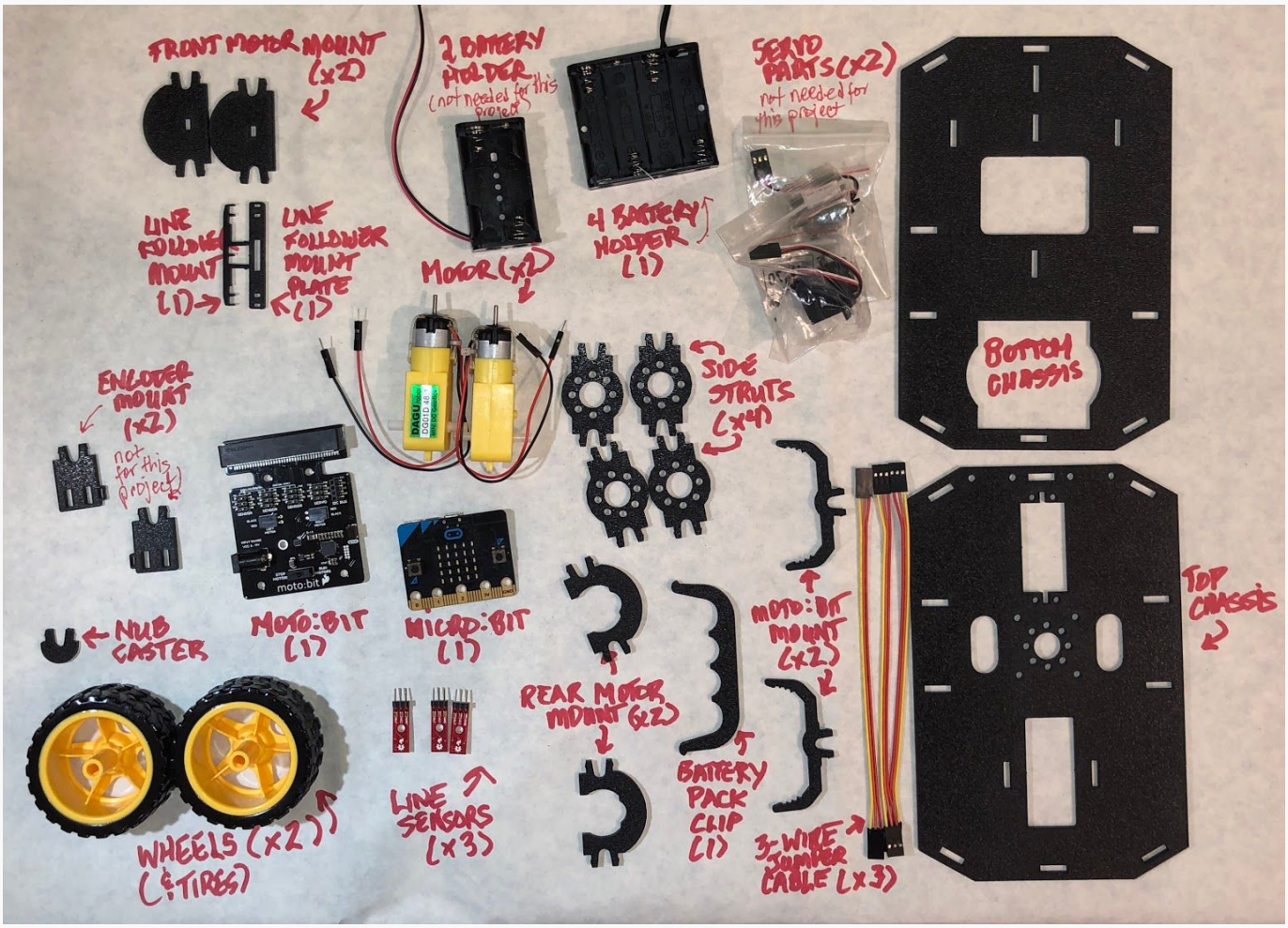
   Suppose the instructions said: "Attach one side strut to each of the four corners of the bottom chassis." (Don't worry, these instructions will be more clear than that, we hope!)

   Even if you didn't know which part was supposed to be the "side strut" (and how could you?), you could figure it out because there's only one part that you have four of!

   It's also a good idea to lay your parts out and count them before starting just in case the manufacturer made a mistake and you're missing a piece -- it's easier if you find this out before you're very-nearly-done and excited to show off your cool project.

5. Another thing I do to help minimize my frustration, especially when I'm working with parts I've never worked with before is labeling them. You may notice that the parts in the picture above are on a white sheet of paper, which serves two purposes: first, it protects my table, and second, it gives me a place to label my parts.

   Before starting assembly, you should read through all the instructions (really). When I'm doing my read-through, I also try to figure out which part is which, and I label them on my paper (along with the quantity). The final result looks something like this:



A note on orientation

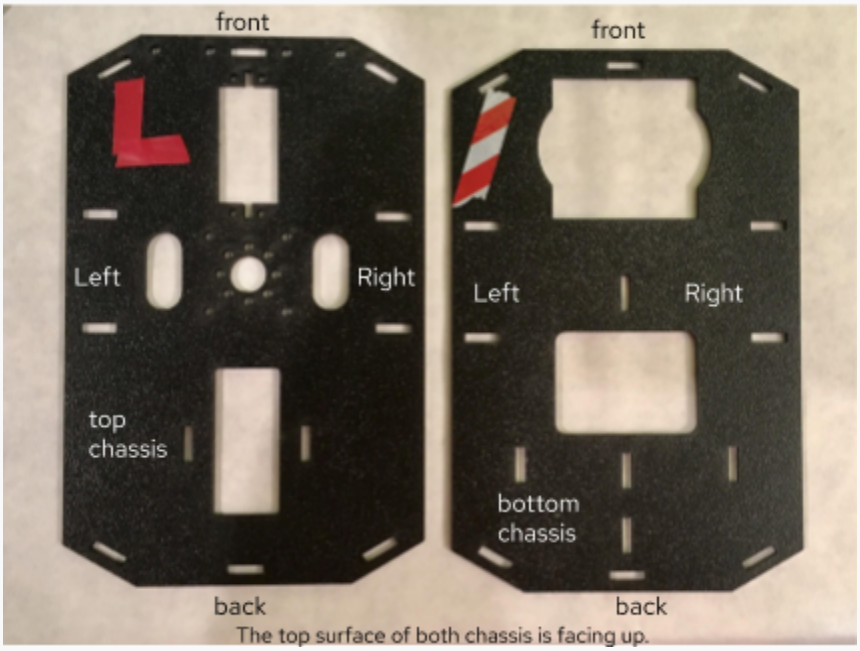
Along with laying out, counting, identifying, and labeling all the pieces in advance, I also like to do whatever I can to make things a little easier when I assemble projects. One way that I do that is by giving myself some obvious orientation clues when I'm working with parts that I might have a chance to put on upside down or backwards.

This can be done in any number of ways -- by marking on one corner of each piece with a pen, adding a sticker, or (my favorite) using some tape. The important thing is to do it consistently -- if you're marking the upper lefthand corner, do that on *all* the pieces. Or if you're using a green mark to indicate the top of the part and red to indicate the bottom, do *that* on all the pieces.

For the robot, the directions reference (see the picture below):

- the Top Chassis and the Bottom Chassis
- the "front" of the robot (the end the front bumper would be on if it had one) and the back
- the "top" of each chassis (the textured side) and the bottom (the smooth side)
- and the Left and Right of each chassis, which is relative to the "front"

I use pieces of tape to mark my chassis -- always on the front, left corner on the top surface. The Top Chassis has a red "L" and the Bottom Chassis has candy cane-striped tape. It doesn't have to be pretty to work.



The top surface of both chassis is facing up.

All this may sound like a lot to do before you even start but if you've ever tried to put something together and you can't find the right part -- or worse, you had the right part, but you used it in the wrong place or put it in upside-down -- this will make sense.

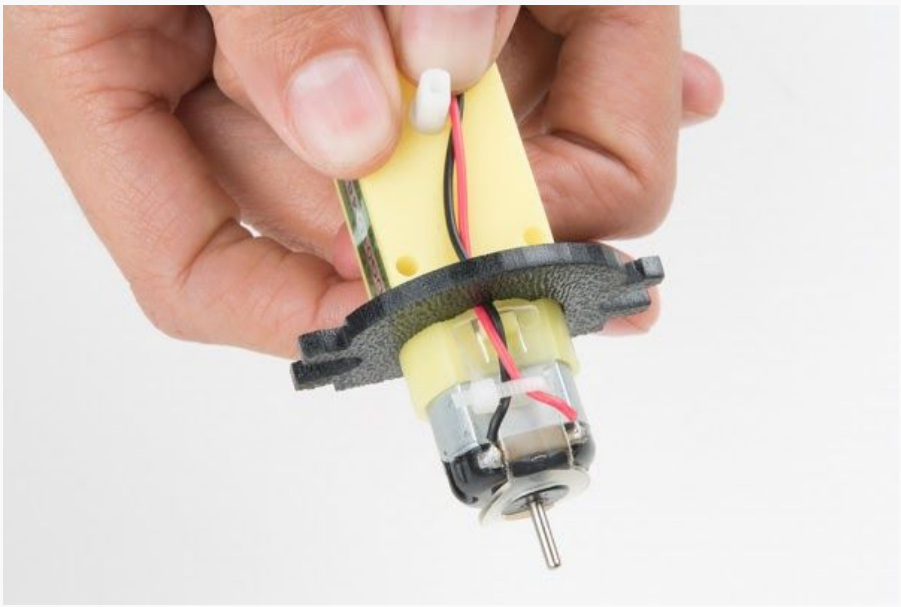
Build the Robot

Put together the motor and wheel assemblies

We're starting with the motor and wheel assemblies, which might also be called a drive train.

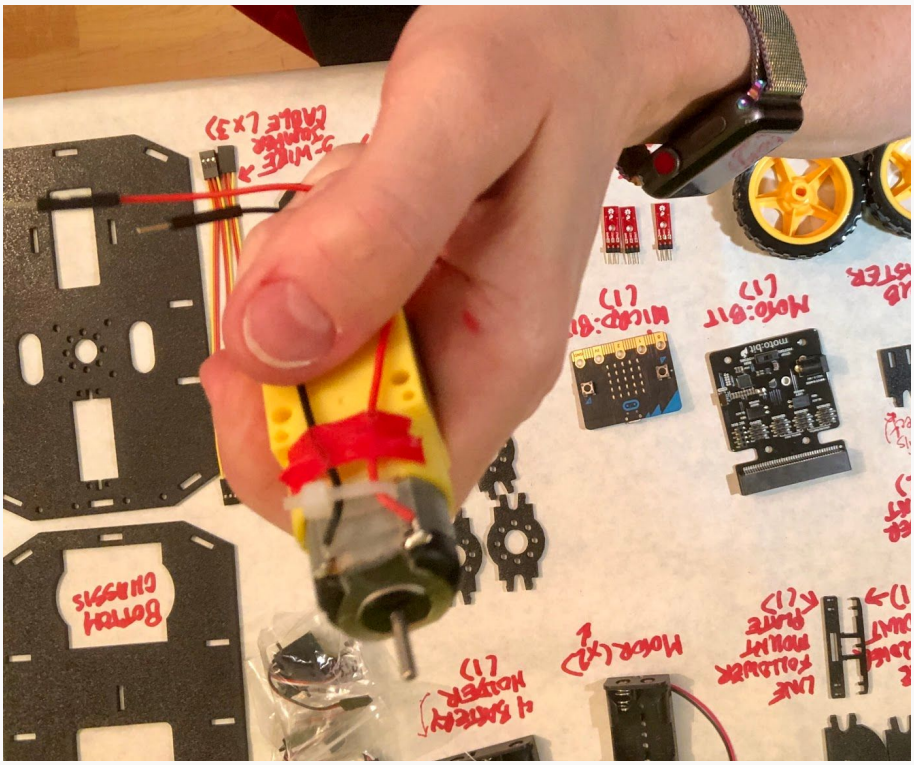
1. Attach the Rear Motor Mount (two options)
   Option 1:
   a. Hold the wires near the middle of the Motor, and carefully slide a Rear Motor Mount in from the side and over the two motor wires. Be careful not to snag the wires, the cable tie, or the clear plastic strap.

   b. Holding the motor wires, gently twist the Rear Motor Mount counter clockwise so that it snaps in place on the motor and the wires are centered in the gap of the motor mount. Again, be sure not to snag the wires under the motor mount.



Option 2:

a. (I think this method is easier) Cut a short (~.75") length of electrical tape and then cut it in half lengthwise.
b. Hold the wires in place near the middle of the Motor, then take one of the two pieces of tape and place it over the two wires on the motor, as shown:



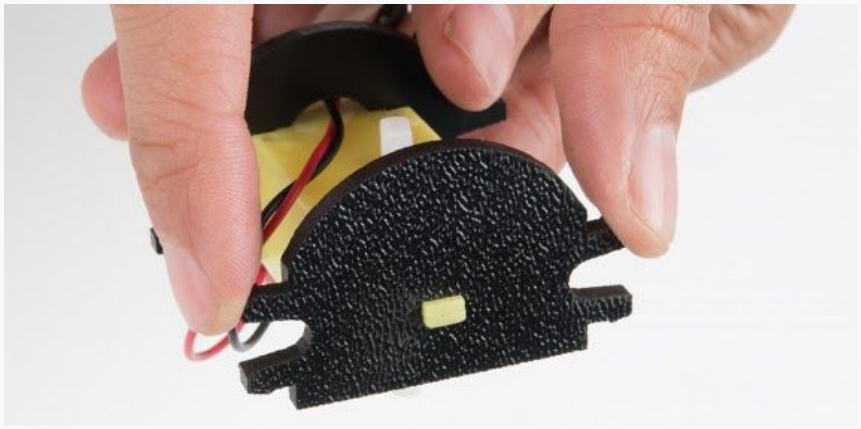
c. Use your channel lock pliers to hold the Rear Motor Mount in position directly in front of the rectangular part of the motor (as shown below) then use your pliers to help you apply pressure straight down on the Rear Motor Mount, like this:

You have to use quite a bit of force, but the Rear Motor Mount should slip over the curve of the Motor to fit snugly, while the wires pass through the small u-shaped cut in the Rear Motor Mount. The electrical tape provides a bit of a barrier as you're applying pressure, because the edges of the Rear

Motor Mount could damage the wires if you were to slip.

2. Using whichever method worked to repeat the process for the second motor, so they both look like this:

3. Slide a Front Motor Mount onto the protrusion on the end of one of your two Motors. Make sure the round parts of both the Front and Rear Motor Mounts are facing the same direction.



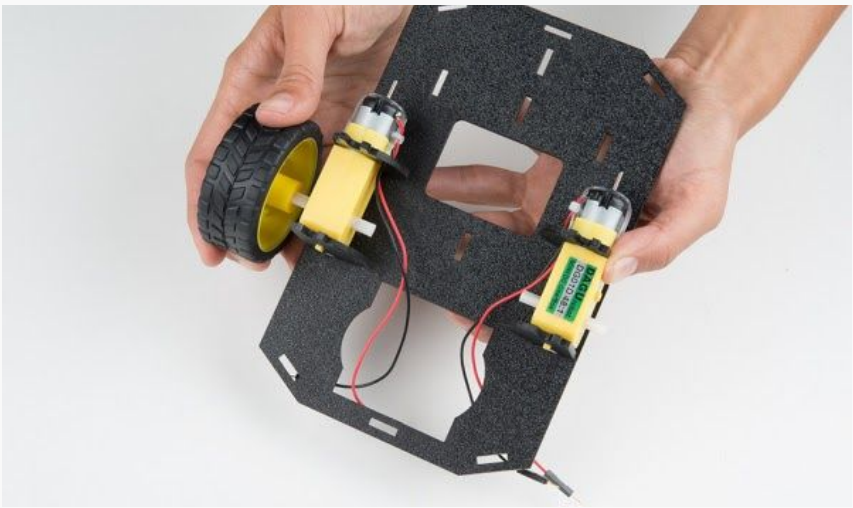
.

4. Repeat the process for the second motor. Congratulations! You have built two Motor Assemblies.

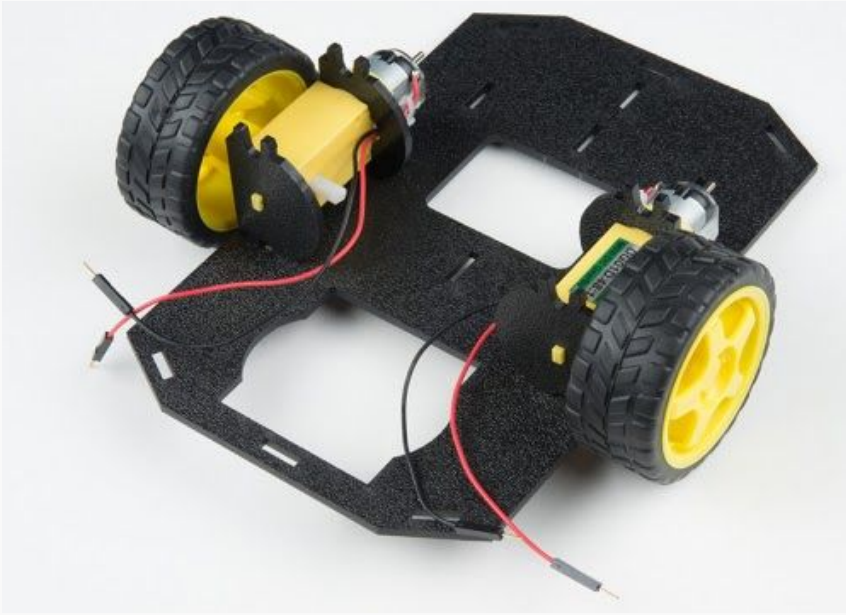
Attach the Motor Assemblies & Wheels to the Chassis

1. Snap one of the Motor Assemblies into the left two horizontal slots of the Bottom Chassis. Make sure that the rounded edges of the motor mounts and the wires are facing toward the center of the chassis and the metal part of the Motor Assembly faces towards the front of the Bottom Chassis. Repeat for the opposite motor.

2. Look at one of the two white pegs (motor shafts) that stick out over the sides of your robot. Notice that it has two flat edges. Slide one Wheel onto the Motor Shaft, making sure to line up the flat edges of the motor shaft with the flat edges of the wheel.



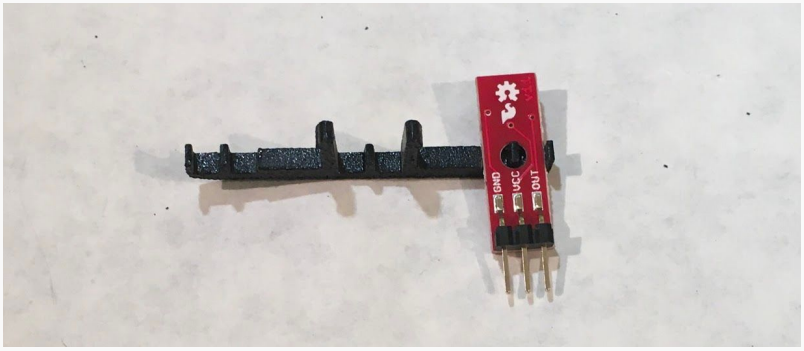
3. Repeat with the other wheel.



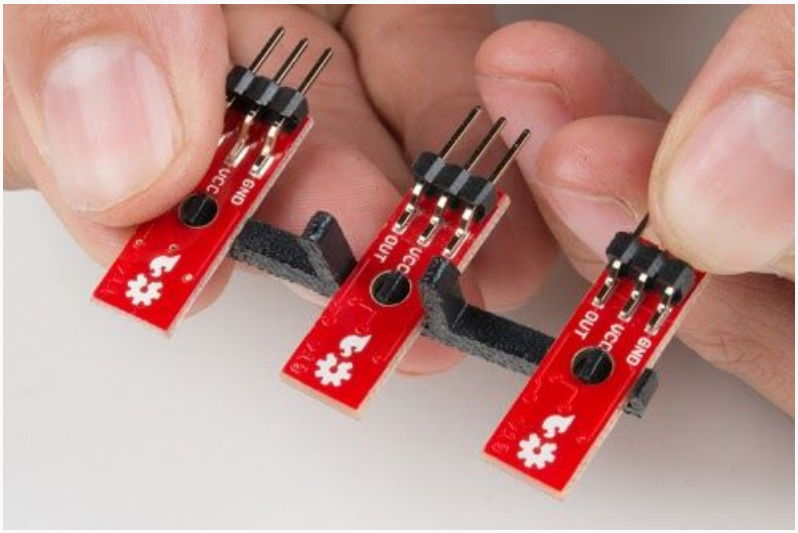
Building & Installing the Line Sensor Array

The Line Sensors emit light from an infrared LED and then detect that light once it has been reflected off an object. This allows your robot to “see” lines!

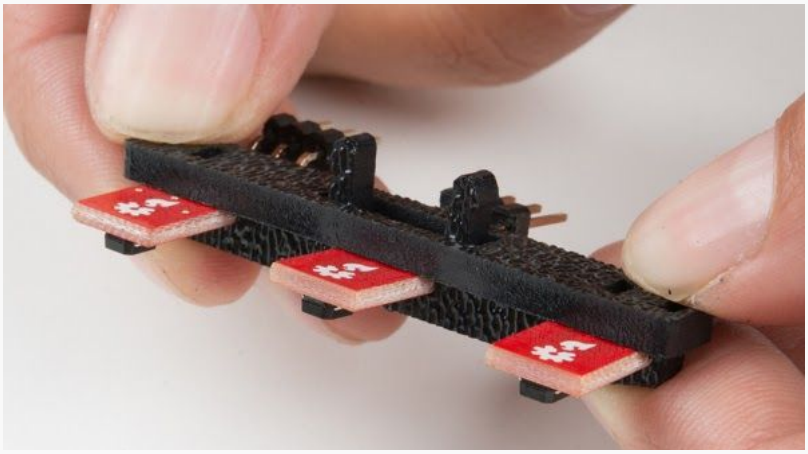
1. Take one of your three Line Sensors and turn it so the sensor faces down and the side with pins faces up. Attach it to the Line Follower Mount by putting the hole in the Line Sensor over the small peg on the Line Follower Mount, like this:



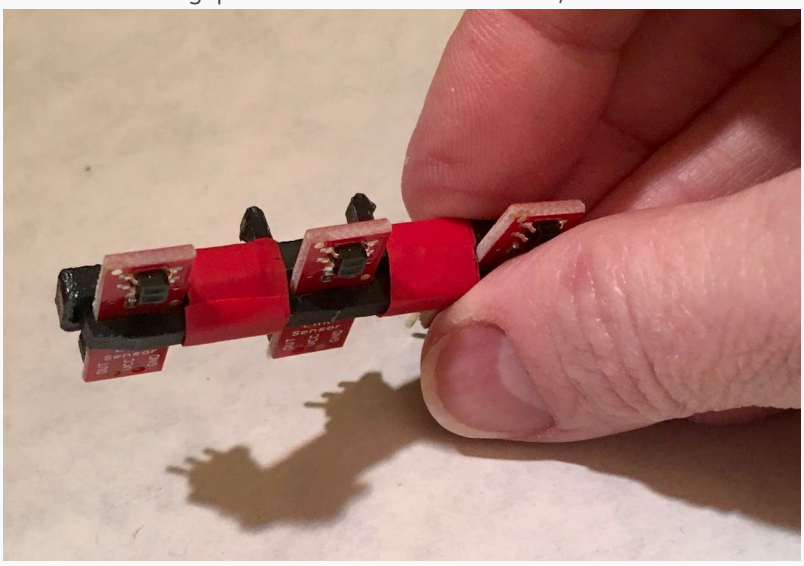
2. Repeat for the next two sensors, so that when all three are set on the Line Follower Mount, you have something that looks like this (notice that the sensors are facing away/down from the prongs of the mount):



3. Place the Line Follower Mount Plate on top of the Line Follower Mount, sandwiching the Line Sensors in between the two. The center clip of the mount should poke through the center slot of the plate.



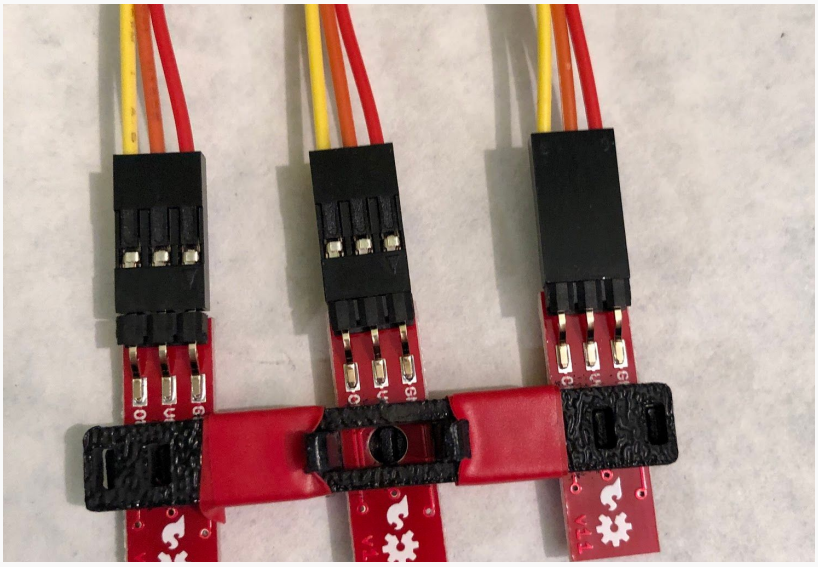
4. The next part is easier if you stabilize your Line Sensor sandwich (which we're going to call a Line Sensor Array from here on out). Take a piece of electrical tape ~1.5" long, cut it in half, then wrap one piece around the entire sandwich in the gaps between the Line Sensors, like this:



5. Next, you are going to connect a 3-Wire Jumper Cable to each of the Line Sensors. Note the color of the wire attached to each pin.

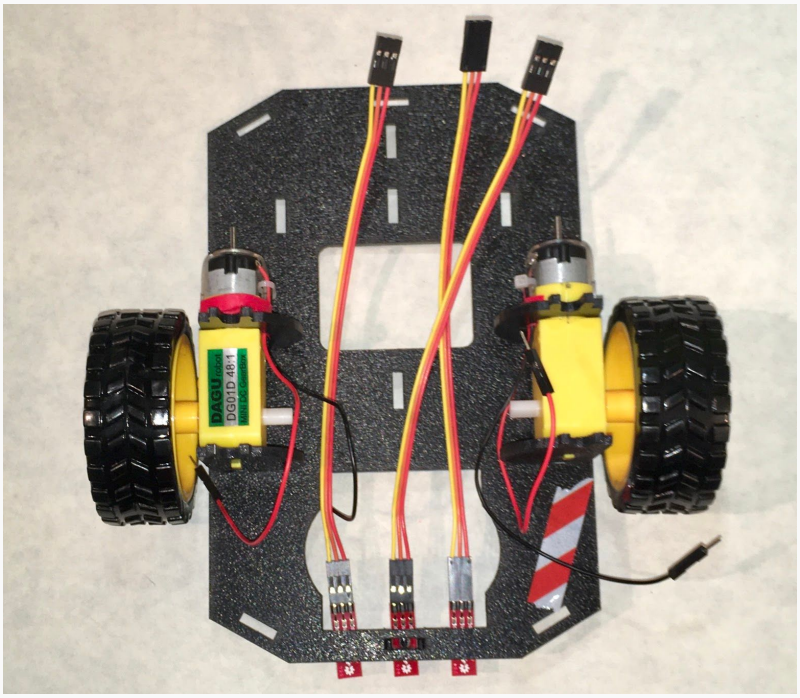
Line Follower Connections	
Jumper Wire Color	Line Sensor pin
Red	GND
Orange	VCC
Yellow	OUT

6. Attach all three cables to the three Line Sensors. When your Line Sensor Array is facing as shown in the picture below (the sensors face down here, as they will in the robot), the yellow wire should be on the left (on the "out" pin) and the red wire should be on the right (ground).



7. To attach the Line Sensor Array to the Chassis, locate the wide, rectangular slot near the front of the Lower Chassis and snap the Line Sensor Assembly in from the *bottom* side of the chassis. When you're done the sensors will be facing the floor and the yellow wires will be on the left.

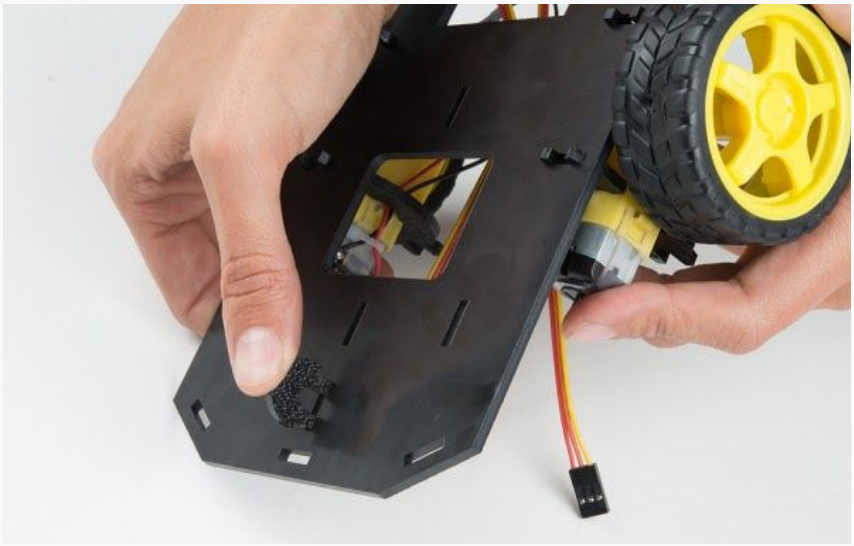
8. Route all three sets of wires through the large hole in the bottom chassis.



Finishing the Body

Yah! All the components are complete, so let's move on to the final chassis assembly,

1. Snap the Nub Caster (go ahead, laugh. It's a funny name) into the slot that's closest to the back of the Bottom Chassis from the bottom side. The Nub Caster, which keeps your robot's tail from dragging the ground, will be on the side opposite the motors and at the other end of the chassis from the Line Sensor Array, like this:

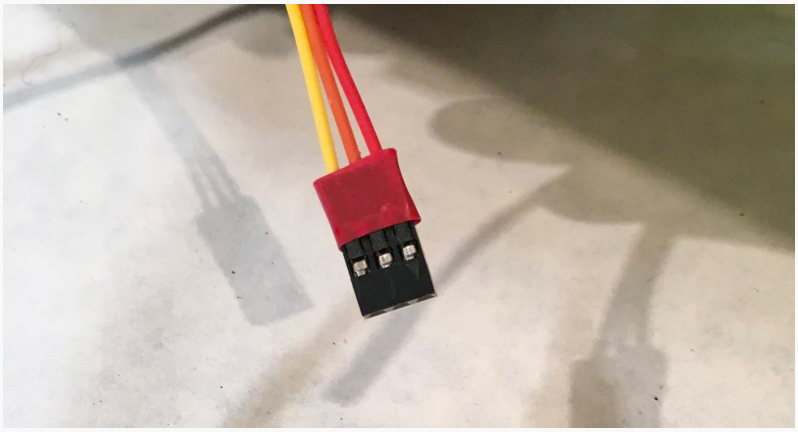


2. Snap the four Side Struts, which will connect the Top and Bottom Chassis, into the diagonal slots at the corners of Bottom Chassis on the opposite side of the Nub Caster (in other words, the Side Struts are attached on top of the Bottom Chassis) .

3. The cables for the motors and the sensor arrays should be sitting on top of the Bottom Chassis, like this:

4. Before routing all the cables it helps to mark the cable bundle for the Middle Line Sensor. The easiest way to do that is to cut a tiny piece of electrical tape (~.25") and wrap it around the piece of plastic that's at the other end of the cable bundle attached to the Middle Line Sensor.

   You only need to mark the Middle Line Sensor cable because it's the only one that might get confused with an identical cable, the Left Line Sensor, because they both get routed through the same hole on the chassis. The Right Line Sensor goes through a different hole, so there's no potential for confusion there.

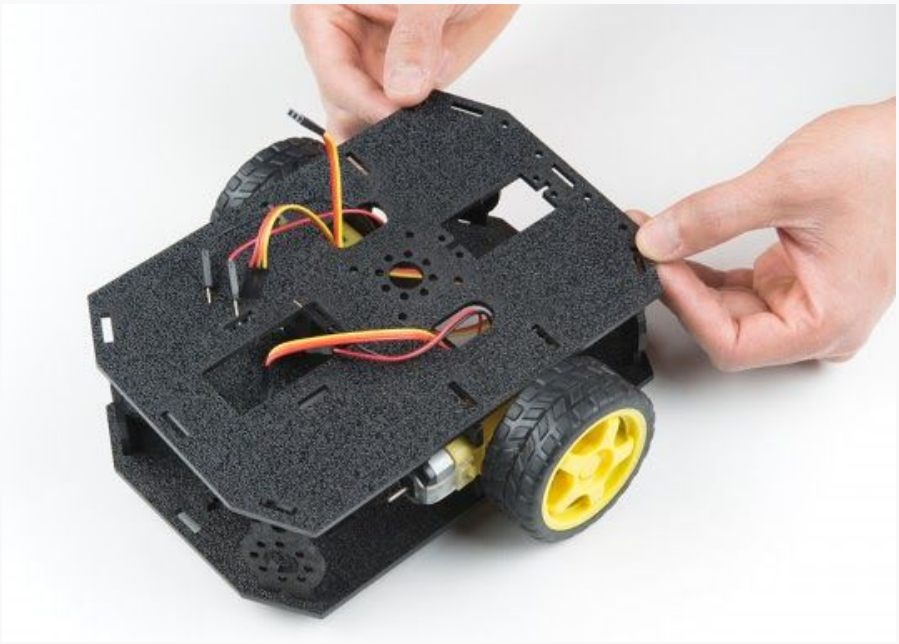


5. Position the Top Chassis over the Bottom Chassis -- but do not snap the two together yet. Make sure that the front sides of each chassis line up.

6. Route the Left Line Sensor cable bundle and the Left Motor wires (red and black) through the left oval slot in the Top Chassis.

7. Route the Right Line Sensor cable bundle and Right Motor wires through the right oval slot in the Top Chassis.

8. That just leaves the Middle Line Sensor cable, which you should also route through the left oval slot. (See why we marked that one with tape? There are two sets of Line Sensor Cable bundles that pass through that hole and the tape helps keep them straight.) Here's what it'll look like:



9. Line up the Top Chassis on top of all the struts, and carefully snap the Top Chassis assembly onto the side struts and motor mounts. Press gently above each side strut individually until they each snap into place.

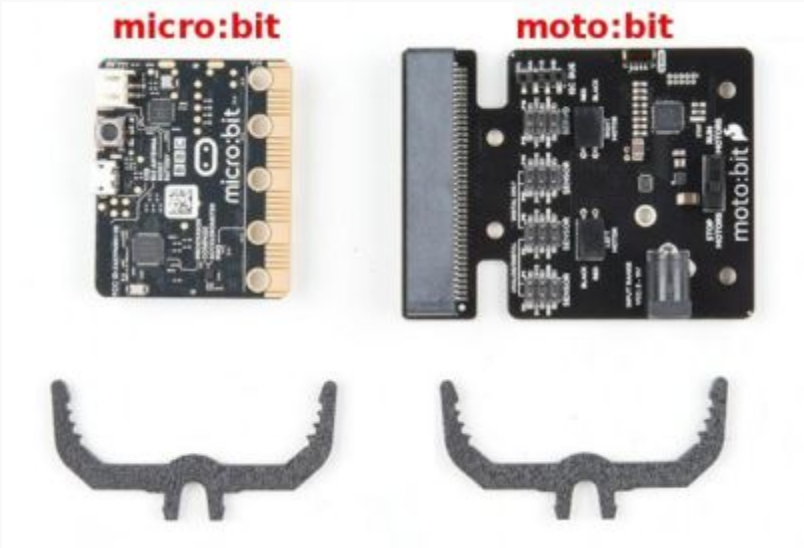
NOTE: If you need to remove the top chassis to change anything, gently pull upward on each side strut

individually. Do not attempt to use pliers or hand tools, or you may end up snapping the plastic clip.

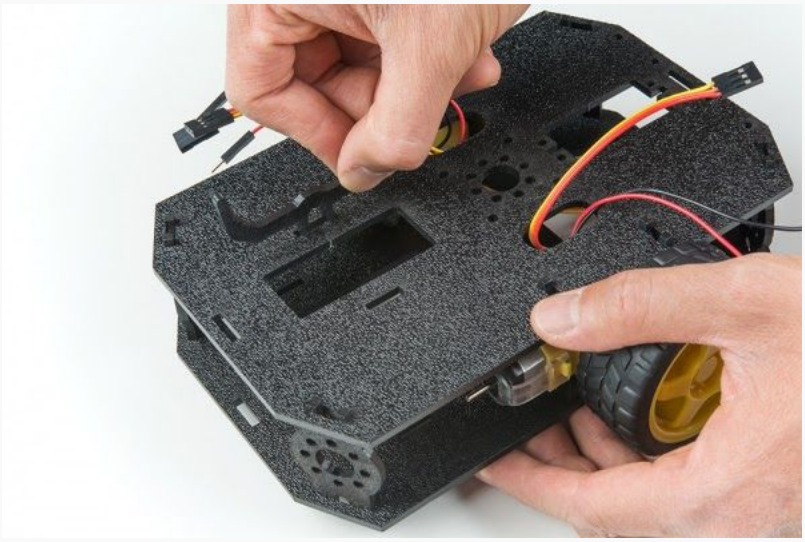
Adding the Brains

In this section, you will add brains of the robot: the micro:bit and SparkFun moto:bit.

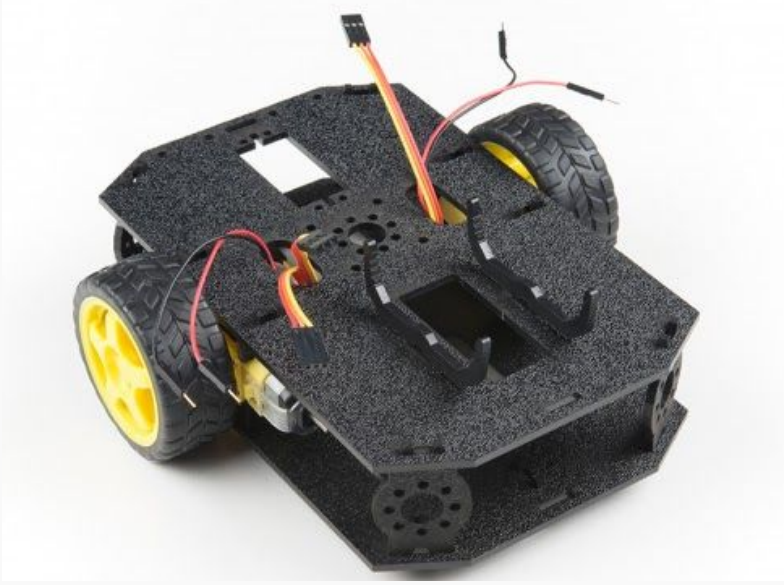
1. Find the micro:bit, the moto:bit and the two moto:bit Mounting brackets.



2. Attach one of the moto:bit Mounts by snapping it into one of the vertical slots in the back of the top chassis near the large rectangular opening.

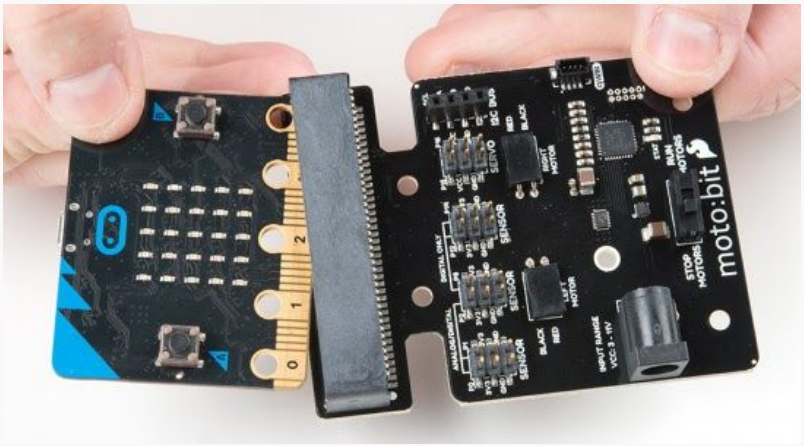


3. Attach the other moto:bit Mount by snapping it into the other slot:

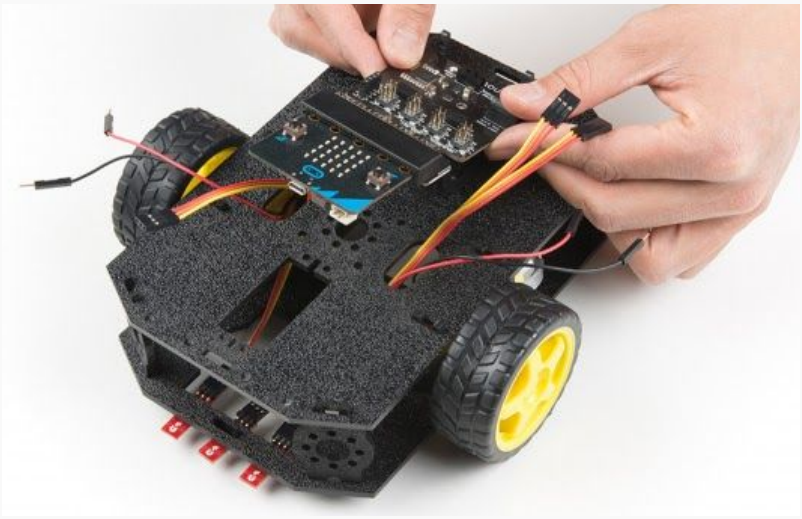


4. Time to insert your micro:bit into the moto:bit.

   Sneaky thing: the “top” of the moto:bit is the side with the pins and chips and the power input on it, but the “top” of the micro:bit is the side that has the LEDs on it (the side *without* the pins and chips and power input). It is possible to put the micro:bit in upside down (ask me how I know ;-) so make sure you have it oriented like this:



5. The moto:bit snaps into the lowest of the notches on the moto:bit Mounts. Make sure the power jack is facing the left side of the robot. Push gently and evenly until it snaps into place.



NOTE: The extra slots in the moto:bit mounts can be used to hold the Arduino Uno or the Sparkfun RedBoard.

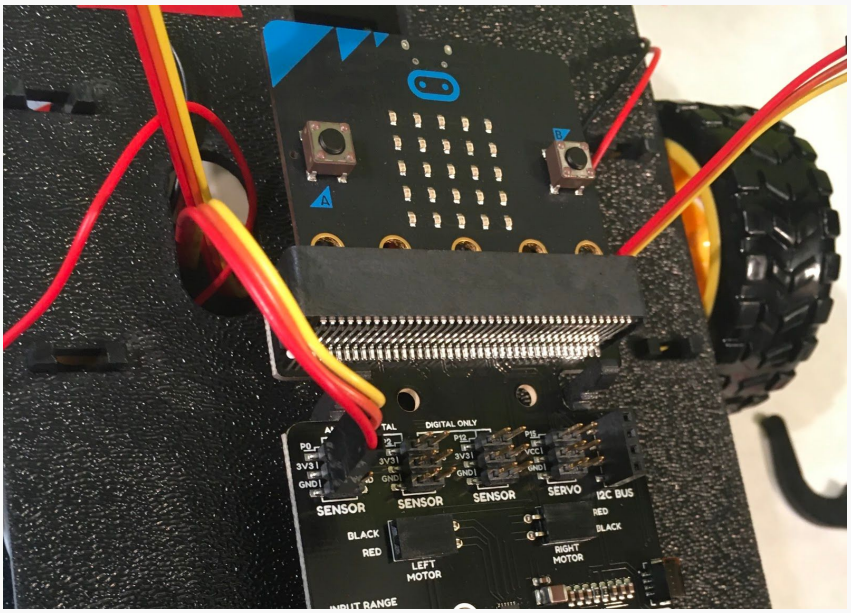
6. Here's what everything looks like before connecting the wires:

7. To hook up the motors, put your robot in front of you with the Sensor Array pointing away from you (as if it were going to drive away from you).

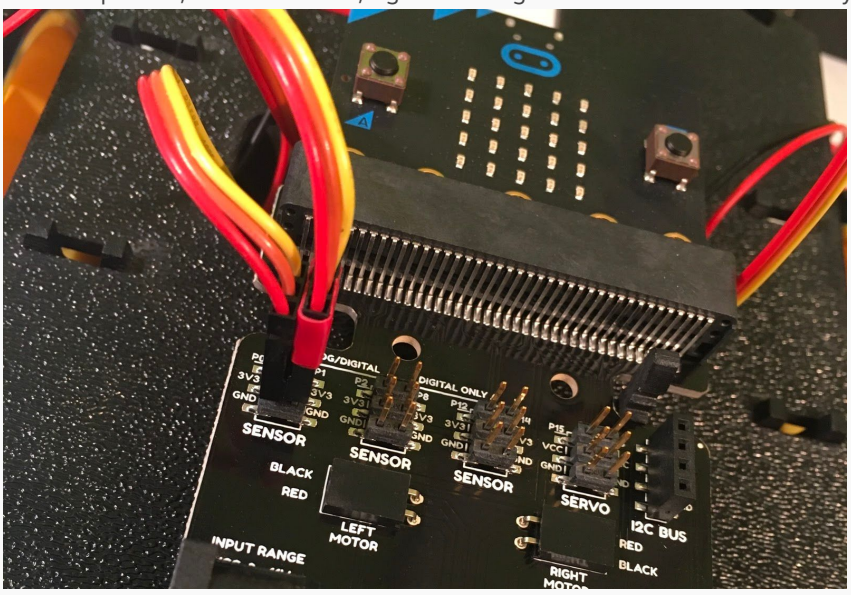
8. While you're making your connections it's especially helpful to be able to keep the robot oriented correctly, so if you haven't done so already, you may want to mark the left front corner of the robot in some way.

9. First, connect the Left Sensor cable bundle (which is the one coming out of the left oval that *doesn't* have the electrical tape on it) to the moto:bit sensor pins labeled P0, 3V3, GND. Since RED was GND when we connected the cable bundle at the sensor end, RED needs to stay GND when we connect the cable bundle at the moto:bit end.

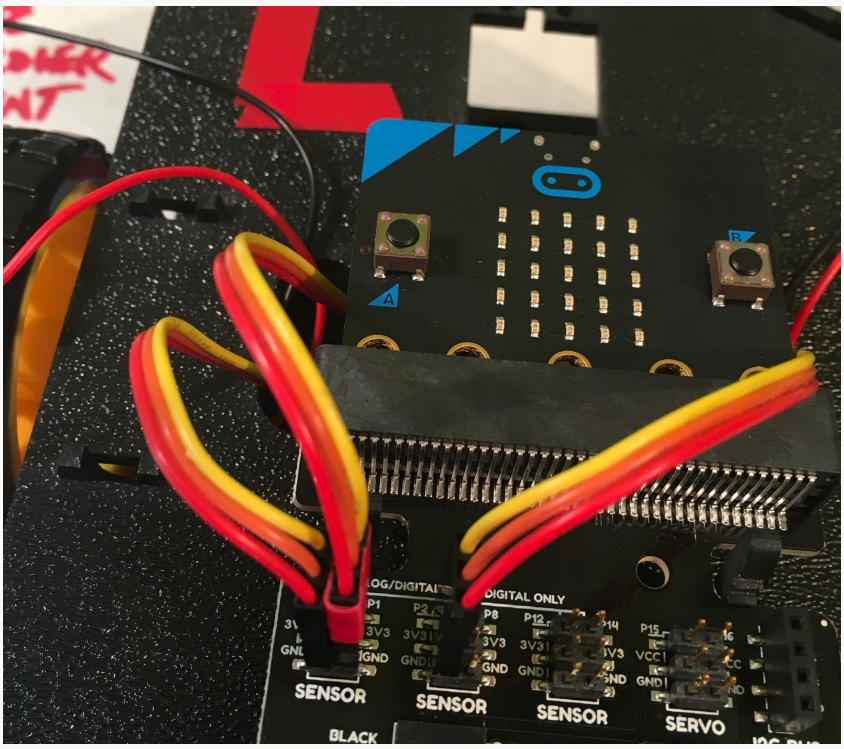
   Once the cable bundle is plugged in the YELLOW wire should be closest to the front of the robot, and the RED wire should be closest to the rear of the robot, like this:



10. Then connect the Middle Line Sensor (the cable bundle that *does* have tape on it coming out of the left hole) with the pins P1, 3V3 and GND, again making sure that the RED wire stays connected to GND:



11. Connect the Right Line Sensor cable bundle (which comes out of the right hole) with the pins P3, 3V3, and GND:

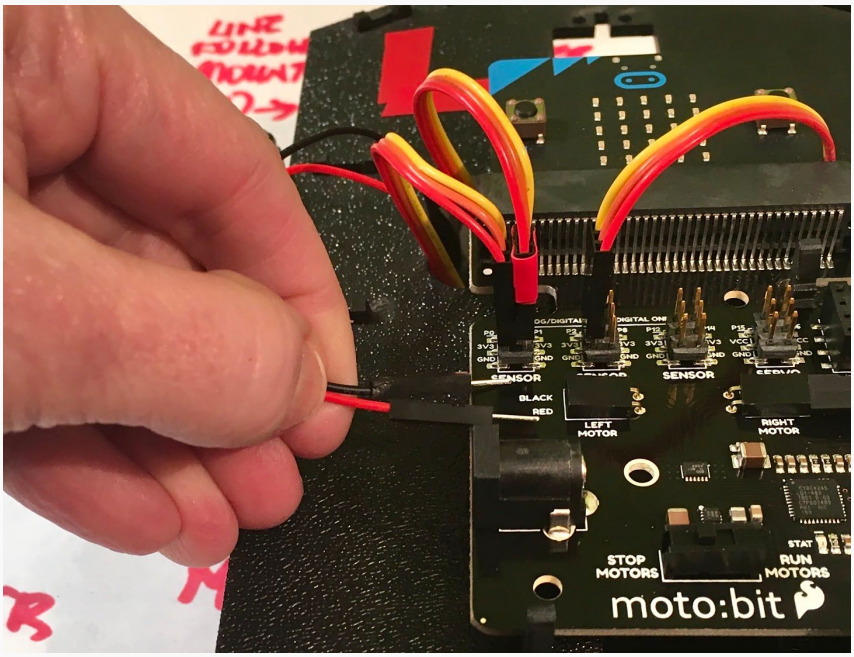


12. Next we're connecting the motors, which is a little weird, but stick with me.

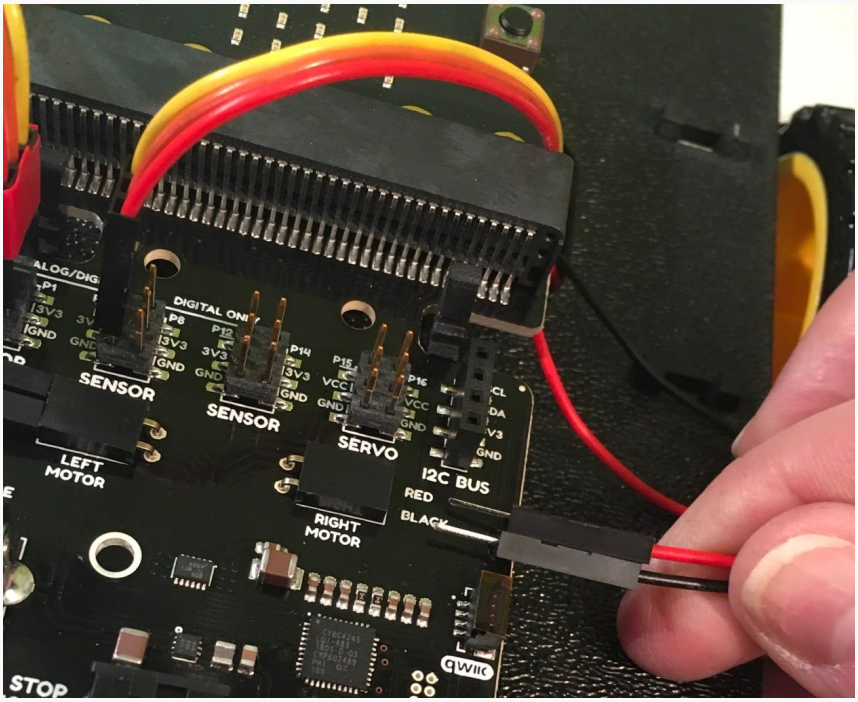
    To make a long story short, there's no way to determine which direction the shaft for each of your two motors will spin before getting it hooked up and running. So, what we're going to do is hook them up the way the moto:bit board indicates for now. Once we have everything else assembled (and we're close -- only the power left to do after this) we'll load some sample code and see if the motors do what we expect them to do.

    If they do -- yah! If they don't, that's ok too, as there is both a hardware and a software fix for the issue.

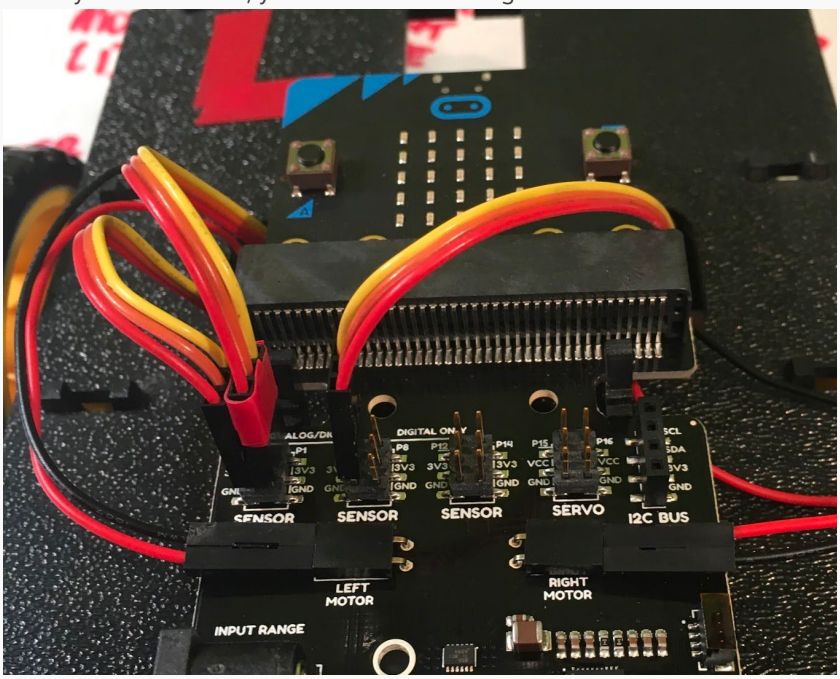
13. Connect the left red and black motor wires (which come out of the left hole) to the left motor connection on the moto:bit board:



14. Connect the right red and black motor wires (which come out of the right hole) to the right motor connection on the moto:bit board:



15. When you're finished, you'll have something that looks like this:



POWER!

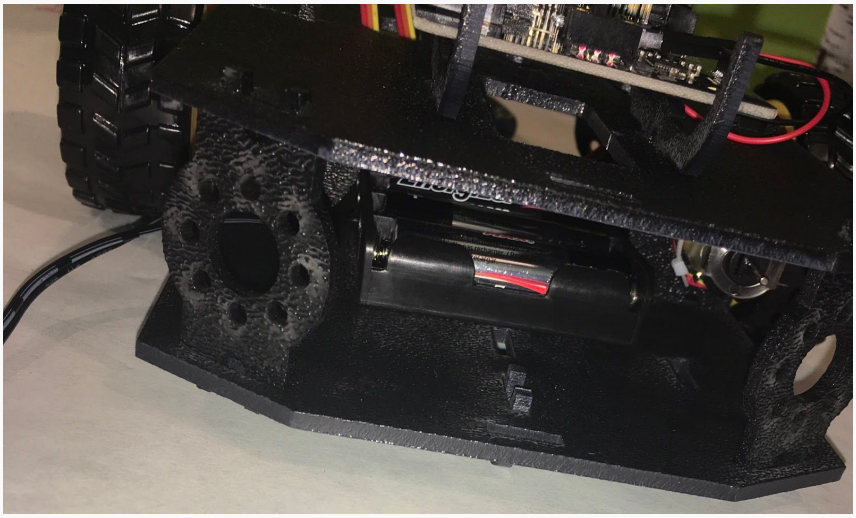
Now we need to give your robot some juice. The power (supplied by 4 AA batteries) goes into the moto:bit, which then powers the micro:bot.

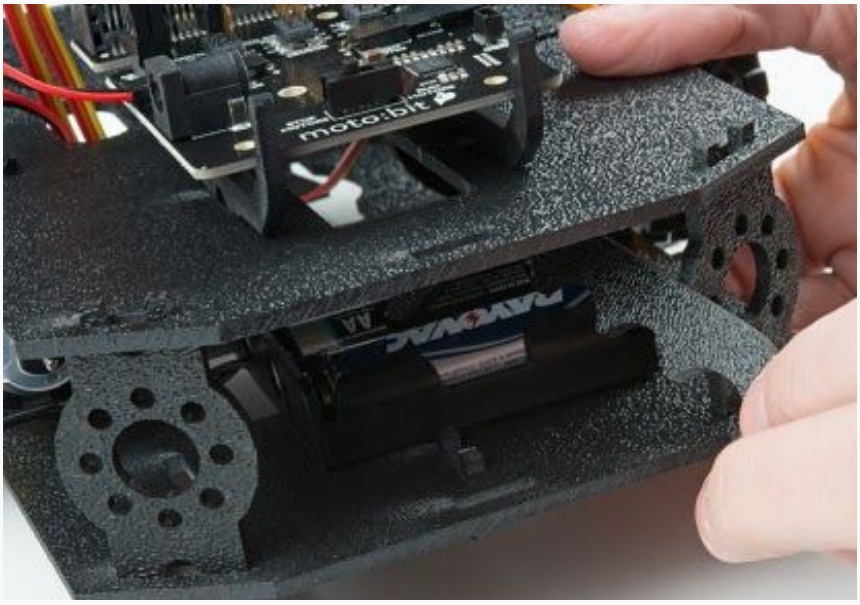
1. You'll need the Battery Pack Clip , the 4 AA Battery Holder, and the provided AA batteries:

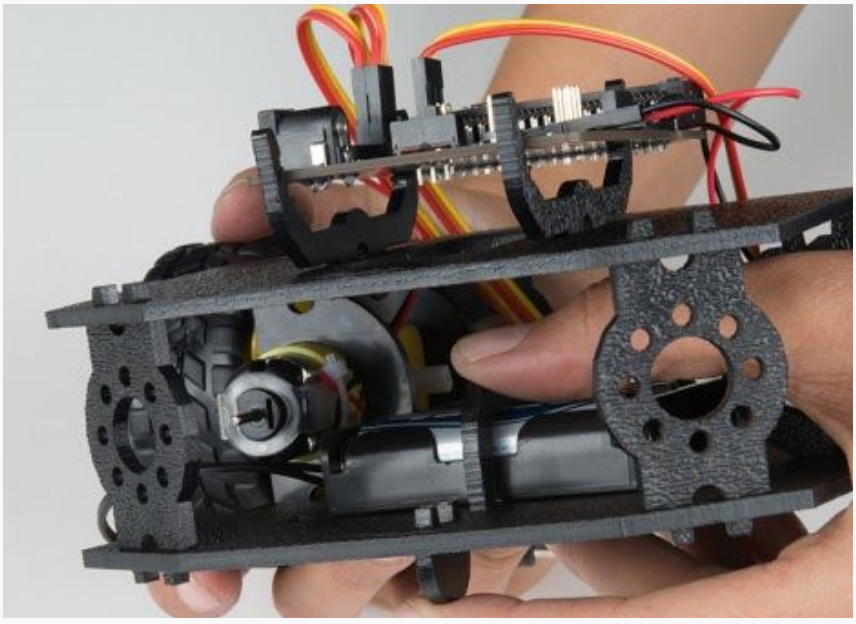
2. First insert the batteries into the Battery Holder, making sure that you face them in the correct direction, as shown in the diagram on the inside of the holder):

3. With your robot in front of you and pointing away from you, orient the Battery Holder (with batteries) as shown, then insert it into the back cavity of the chassis:

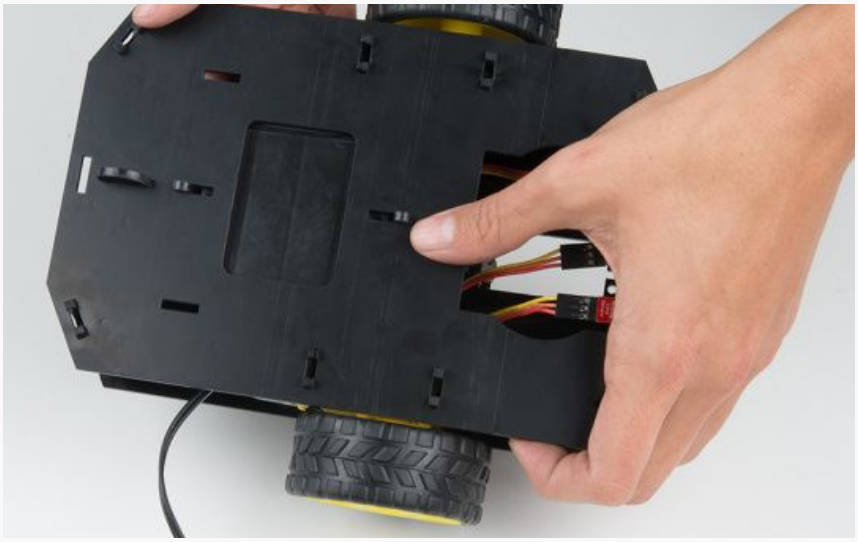
4. Position the Battery Holder so that the barrel jack cable comes out on the left side of the robot and the Battery holder sits on top of the hole, like this:



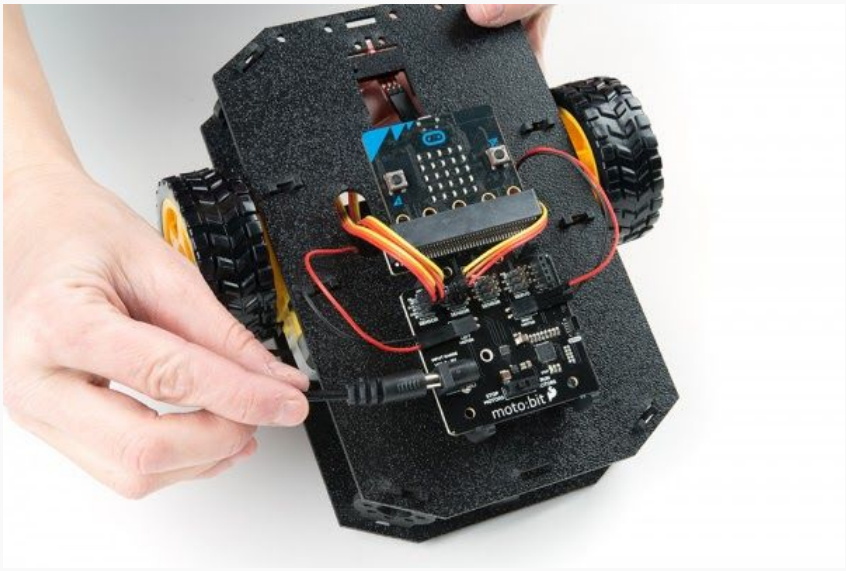
5. Insert the Battery Pack Clip on top of the battery pack (with the open end facing down towards the batteries), twist and position the clip so that it rests on top of the battery pack.
6. 



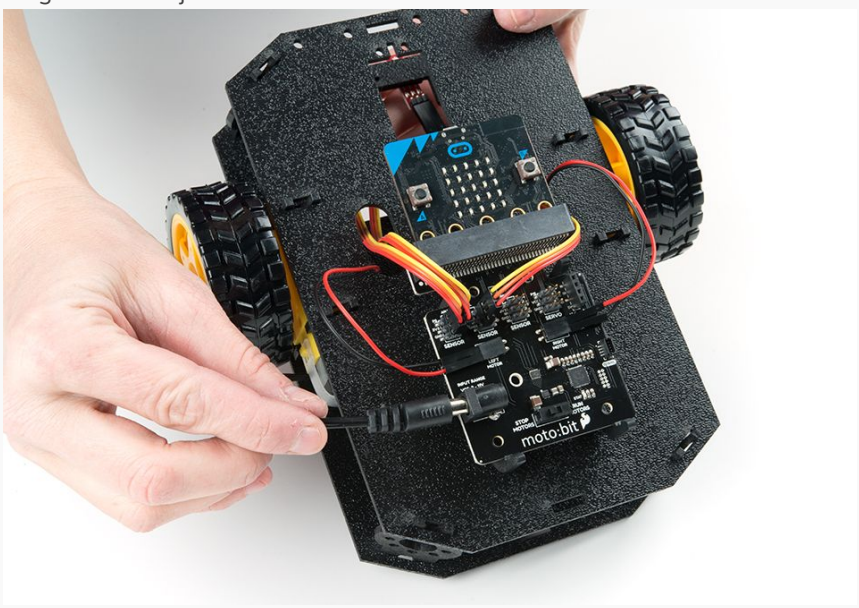
7. Push the clip down into the vertical slots in the Bottom Chassis so it snaps in place.



8. Route the barrel jack cable out of the left side of the chassis and up to the moto:bit.



9. Plug the barrel jack cable into the barrel connector on the side of the moto:bit carrier board.



Pro tip: there's a switch on the moto:bit that you can turn off when you're not using your robot to save power. **Even more pro tip**: even when you turn the power switch off there's still a small amount of power going to the moto:bit, so to conserve as much battery life as possible, unplug the barrel jack if you're not going to be using your robot for a while.

Instructions for changing the batteries can be found in the "Additional Information" section.

Get the Robot Running

One of the (many) cool things about the micro:bot (and by extension, your robot), is that you can learn to control it (to program it) in several ways. We're going to use MakeCode, which is a open source, block-based language: if you're familiar with Scratch, it's very similar.

MakeCode can be run in your browser or in an app (Android, iPhone or iPad). We're going to work with the browser-based version, as it's the easiest to set up.

Using MakeCode

If you've never used MakeCode, start by visiting one of these two sites to learn the basics: how to write programs for the micro:bit, and how to connect to a micro:bit and transfer your code so that it can run.

- SparkFun's guide: https://learn.sparkfun.com/tutorials/getting-started-with-the-microbit?_ga=2.138146608.1695833731.1587361358-1522773540.1573599070#using-makecode -- has more detail about what the different tools in the MakeCode Editor do and how the in-browser simulator works. A better choice if you prefer to read All The Directions before getting started on a project.
- The Micro:bit site guide: https://microbit.org/get-started/first-steps/set-up/ -- has a couple short videos and some brief information about how to get started with the micro:bit. If you prefer to jump right in and figure things out as you go this might be more up your alley.

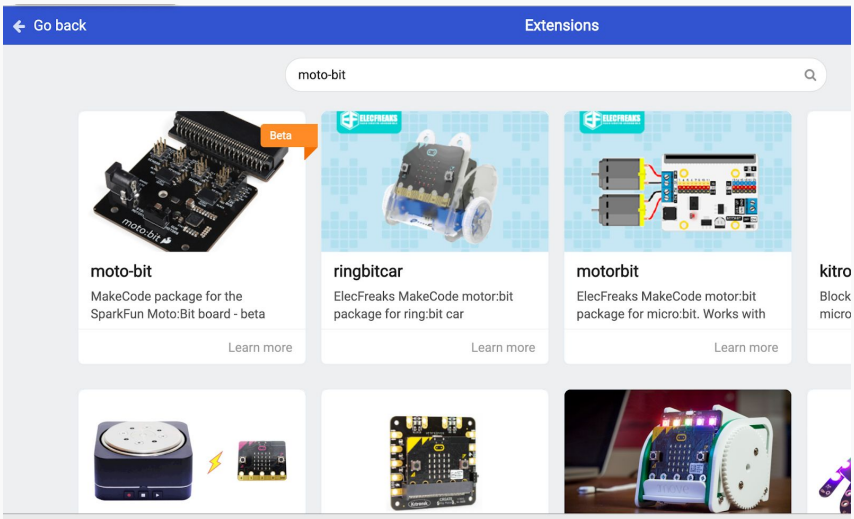
No matter which introduction you choose, though, you'll need to complete one additional step in order to access your robot's full potential...

Install the moto:bit Extension in MakeCode

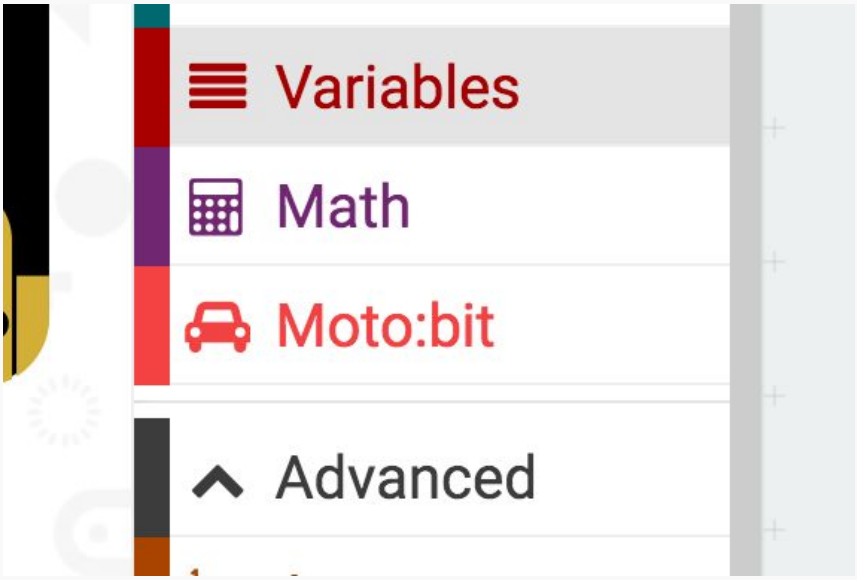
In order to take advantage of the motor:bit's full functionality, SparkFun has created an "extension," which is a file that contains code to -- you guessed it -- extend the functionality of the original code base.

1. To install or add a new extension to your MakeCode toolbox (the list of different block groups), click on "Advanced" and then on "Add Extensions." This should be the last item on the list.

2. From here you can search for "moto-bit," and it should show up as a public extension in the list. Go ahead and click on it.



3. This will add all of the blocks to your toolbox. When you look at your toolbox you should see:

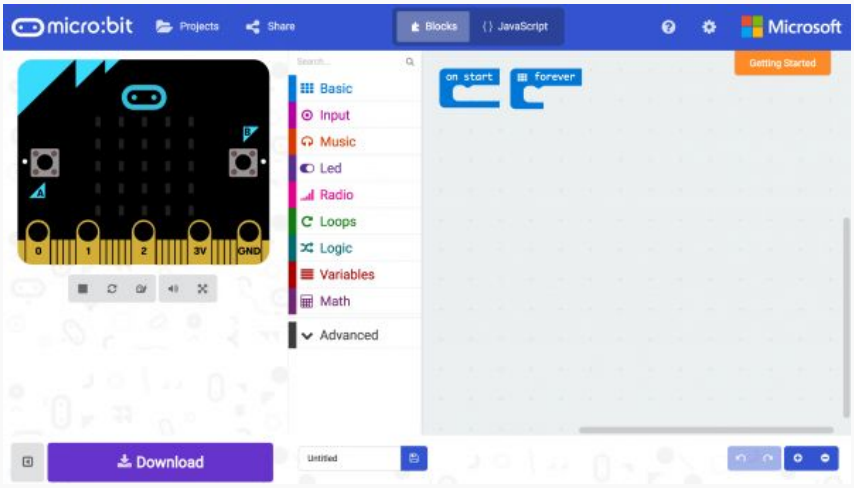


4. Great! You have now installed the moto:bit extension and are ready to use the board as well as the components that come in the micro:bot kit.

   NOTE: For every new MakeCode project that you make, you will have to load extensions over again. Not a big deal, but important to remember.

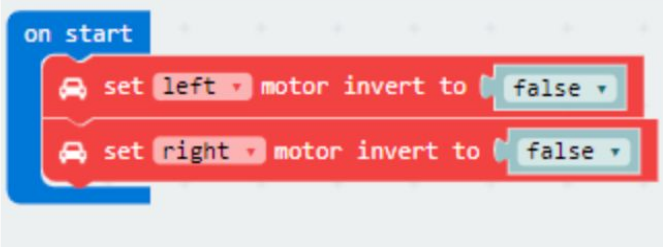
Load the sample code to test your robot

1. We are going to use Microsoft MakeCode to program the micro:bit. Please open a browser window and navigate to makecode.microbit.org. This should open the MakeCode environment that you used to install the moto:bit package in. (If you need to install the moto:bit package, follow the instructions in the Installing the moto:bit Package in MakeCode section of this tutorial.)



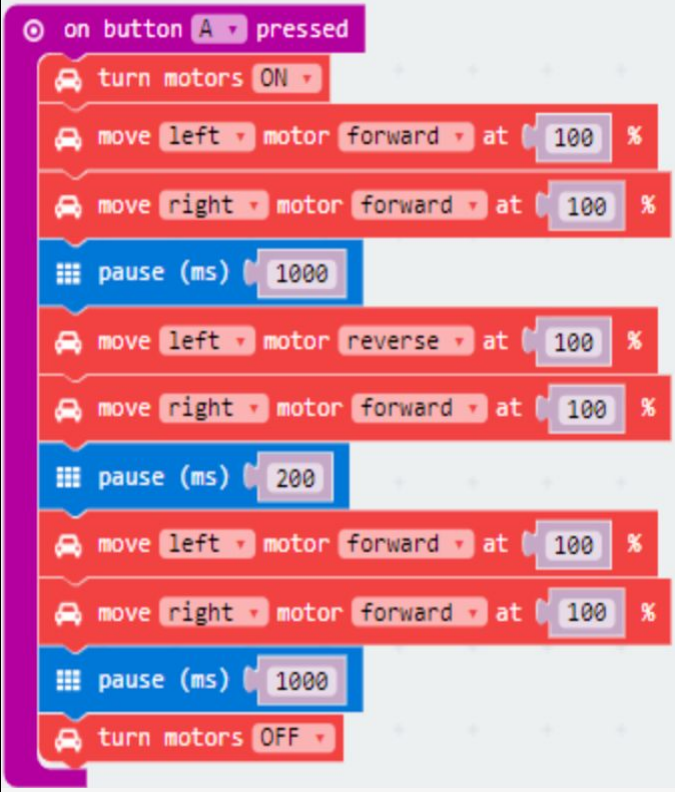
2. Or (a shortcut):, if you click on this link it will open the code we're going to use to get your robot running in a new browser window: https://makecode.microbit.org/#pub:_ce5RocdRcFPM

3. Let's look at what the code is doing



On Start
The code starts by setting each motor channel in the **`on start`** code block. Depending on how the motors are wired to the moto:bit, your micro:bot may drive in the opposite direction than what you would expect. The **`set ____ motor invert to ____`** provides the option to switch the motor wires in code without physically rewiring the motors to the moto:bit. Set the **`left`** or **`right`** motor that you would like to invert. If you want the motor to drive in the opposite direction relative to your setup, select either **`true`** or **`false`**.

Don't worry about this section right now -- we'll come back to it when we see how your motors are working.



On Button Pressed
The **`on button __ pressed`** code block is an event function. Any code blocks that are placed inside of the block will only execute when that event happens. In this block, when the A button is pressed on your micro:bit your robot will start moving. When it gets to the end of the program it will stop again until you press the A button again.

Turn Motors
The **`turn motors __`** block gives you the ability to turn the motors **`ON`** or **`OFF`** in software. They are naturally **`OFF`**, so anytime you want to drive motors, you need to set them to **`ON`**. A good practice is to turn them **`OFF`** when your robot completes its task or should not be moving for long periods of time, this saves batteries and wasted power consumption.

Move Motors At
The **`move ____ motor _____ at _ %`** block is the basis of the moto:bit software package. You use this block by selecting the values in the drop down menu:

- which motor you want to control (**`RIGHT`** or **`LEFT`**)
- its direction (**`FORWARD`** or **`REVERSE`**)
- the throttle / power percentage you would like the motor to spin at (**`0-100%`**)

For your bot to move forward you need to have both motors drive in the same direction.

To turn, or pivot, you set the motor directions in an opposite configuration. (The middle two lines that say **`move _left_ motor _reverse_ at 100 %`** and **`move _right_ motor _forward_ at 100 %`** in the sample code shown is a pivot turn.)

Pause
To understand what **`pause`** does, it's useful to remember that once you turn a motor on it won't stop running until

	you turn it off. The other thing that's helpful is to know that this pause (ms) ____ doesn't mean "stop what you're doing for this long" (like Pause on a music player would). Instead, it's more like "wait this long before doing anything else." So, the **ON** and **move _____ motor _____ at _ %** blocks sets your motors going, then the **pause** block makes your robot wait for the length of the **pause** before doing anything else. And while it's waiting for the time allocated to the **pause** command to pass, it just keeps on doing what it was doing before. So, if you want your robot to drive forward for 1 second, you set the robot's motors to drive forward and then pause for 1000 milliseconds (= 1 second) and then have the motors do something else, like stop.

4. Ok, now that you've got the code in the MakeCode editor, we need to transfer it to your robot. First, click the Download button to download the code, which will be a .hex file, to your computer.

5. Before transferring the code to the micro:bit, make sure the switch on the moto:bit is set to "off:.

6. Then, use the microUSB cord to connect the micro:bit to your computer. The robot should show up as a type of device (like a drive) in your Finder.

7. To transfer the code to your robot, drag the .hex file onto your robot in the finder. The yellow light on the back of the micro:bit will flash rapidly while the code transfers.

   NOTE: If you're using a recent version of the Google Chrome or Microsoft Edge web browsers, you can send programs direct from the online editor to your micro:bit using WebUSB (https://microbit.org/get-started/user-guide/web-usb/).

Take your robot on a test drive!

If your code is loaded and your robot is assembled, it's time to take your robot for a spin --

1. Unplug the USB cable (you don't want to drag your computer off the table)
2. Make sure you have plugged in the barrel power jack (and that you have batteries installed).
3. Make sure the STOP/RUN MOTORS switch on the moto:bit set to RUN
4. Put your robot on the floor (it would stink to have your cool, new robot drive right off the edge of a table!)
5. Buckle your seatbelt and fasten your helmet and press Button A.

What we expect will happen is that your robot will drive forward for 1 second, pivot, and then drive forward for another second before stopping.

What might happen is that your robot drives *backward* instead of forward **or** it might spin in a circle! -- *what!?*

No, you didn't do anything wrong; sometimes wiring is weird.

If you want to go ahead and *fix it* skip to the fixing motor issues section. (No judgement!)

If you'd like to learn a little more about how motors work (and how engineering happens) before fixing it, read on:

> Motors work by sending current around a coil of wire, creating a magnetic field (30-sec video showing this in action: https://www.youtube.com/watch?v=AgZHqfIBkUI). This then interacts with magnets in the motor, causing a push / pull effect. If you do that with proper spacing and timing, then you can spin a shaft (more

about how direct current (or DC) motors work, with some cool demonstrations: https://www.youtube.com/watch?v=onjFFoOC_yk).

When we hooked up the motors to the moto:bit board, we did so with the assumption that the **red** wire is the positive (**+**) and the **black** wire is the negative (**-**), which is the standard (also called the "convention") and that the current would flow through the coil inside the motor from the positive (**+**) **red** end to the negative (**-**) **black** end. That would cause the shaft to spin in one direction (clockwise, normally). The motor also has a gearbox that both turns the clockwise spin 90° (so the drive shaft sticks out the side of the motor rather than the front -- much more handy for mounting wheels) and turns a very fast, but fairly weak spin into the slower but more powerful spin that's needed to make your robot go.

So, what happens if sometimes the factory makes the **black** positive (**+**) and the **red** negative (**-**) -- and yes, this sometimes happens -- and you wire the motors following the markings on the moto:bit board, which was engineered based on the convention that **red** wire is always positive (**+**) and **black** wire is negative (**-**) ?

Well, if you get two of these "reverse" motors then you get a backwards-driving robot. If you get one normally wired motor and one reverse-wired motor, then your robot will spin in a circle (which is kind of cute, but not what we're looking for).

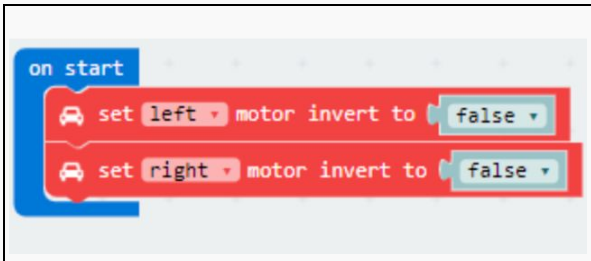
Fixing Motor Issues

There are two ways to fix a reverse-wired motor: in the software or in the hardware. We'll look at both (and you can try them both), but if I have a choice, I'll usually use a hardware fix because there's less chance that it will interfere with something else I might want to do in my code later on.

But if you're not sure whether there's a reverse-motor situation happening, you can use the software fix to test that hypothesis out. If it turns out that you have (one or more) reverse motors, then you can make a more permanent fix with hardware. Because it's useful to diagnose a problem before making fixes, we're going to look at the software fix first.

Reversed-Motor Software Fix

1. If you closed it, open your MakeCode window again. Remember that first block of the sample code with the **`set ____ motor invert to ____`** in it?



On Start
The code starts by setting each motor channel in the **`on start`** code block. Depending on how the motors are wired to the moto:bit, your micro:bot may drive in the opposite direction than what you would expect. The **`set ____ motor invert to ____`** provides the option to switch the motor wires in code without physically rewiring the motors to the moto:bit. Set the **`left`** or **`right`** motor that you would like to invert. If you want the motor to drive in the opposite direction relative to your setup, select either **`true`** or **`false`**.

2. If your robot is spinning in a circle:
   a. Try setting either of the two motors to **`true`** instead of false. It doesn't matter which you choose.
   b. Download the code and transfer it to your micro:bit like you did before.
   c. Press the A button and see what happens. If your robot now goes backwards then your random choice of which motor to set to **`true`** was the wrong one. In other words, you reversed the motor that was correctly wired.
   d. No problem, though, just repeat the process, returning that motor to **`false`** and setting the other motor to **`true`**. Then download your code to the robot and try it out.

    e. Hopefully your robot is now driving in the right direction? Congratulations! You correctly identified the reversed motor, and you can (if you'd like) fix the motor in the hardware. Don't forget to change the software back to its original state!
3. If your robot is running backwards:
    a. Try setting both motors to `true` instead of false.
    b. Download the code and transfer it to your micro:bit like you did before.
    c. Press the A button and see what happens. If your robot now goes forwards, congratulations -- you got two reversed motors, and you figured it out!
    d. Now you can (if you'd like) fix the motor in the hardware. Don't forget to change the software back to its original state!

Reversed-Motor Hardware Fix

1. If your robot is spinning, you have to figure out which motor is reversed (which is where the software fix comes in handy).

   Once you know which motor is reversed, the hardware fix is to reverse the wires where they plug into the moto:bit.

   Example: Let's say your robot was spinning, and you used the software fix to figure out that your left motor is reversed. To fix it in the hardware, you'd plug the **red** wire into the pin that's labeled **black** pin on the moto:bit on the left side and the **black** wire into the pin labeled **red** on the moto:bit on the left side. (Don't forget to go back into your code and reset the left `motor invert to false.` Then you would download your new code and re-test it.)

2. If your robot is driving backwards, it's the same process as for the spinning robot, only you have to reverse the sets of wires on both the right and left sides of the moto:bit (and change, re-download, and test your code).

   NOTE: For other troubleshooting tips, the full kit content listing, the master connection list, and information about changing the battery, see Additional Information.

Congratulations! You have an amazing Co.Lab robot -- we can't wait to see what you do with it.

What's next?

More fun things to do with your Co.Lab robot kit! Keep an eye on this space as we'll be releasing more activities over time!

Additional Information

Full Kit Contents List

Red Hat Co.Lab Robot Kit (red.ht/robot-kit) includes:

- 1x SparkFun moto:bit (Qwiic) - Carrier board multiple I/O pins capable of hooking up servos, sensors and other circuits.
- 1x Shadow Chassis - The frame for your robot.
- 3x SparkFun Line Following Sensor - Three sensors for detecting lines and nearby objects.
- 2x Hobby Servo Motors - Tiny motors that provide control for positioning
- 2x Wheel – 65mm (Rubber Tire, Pair) - Wheels to attach to the Hobby Motors.
- 3x Jumper Wire – 3-pin, 6" - Wires to connect the line sensors to the moto:bit.
- 2x Hobby Gearmotor - Motors for driving the robots wheels.
- 1x 4xAA Battery Holder - Battery pack for powering the micro:bit and the motors.
- 1x micro:bit - The brain for your robot
- 1x Micro-B USB Cable - Used to connect the micro:bit to a computer
- 4x AA Batteries - For power
- 2x Co.Lab Stickers - To decorate your robot
- 1x Red Hat Sticker - To decorate your notebook or whatever else you want
- 1x Co.Lab Postcard - More information about open source robots
- 1x Battery Holder for AAA batteries (you don't need this now)
- 2x AAA Batteries (you don't need these now either, but hang on to them)

Master Connection List

This table shows you how things are connected. On the left side of the table is the board (the moto:bit in this case), in the middle is the color jumper wire, and in the right column is the "far end" connection (either one of the motors or one of the Line Sensors). This is a kind of shorthand for the step-by-step instructions that tell you exactly where each end of each connection goes, so sometimes instructions will give you a table like this rather than step-by-step instructions.

Left Line Sensor Connections:

SparkFun moto:bit Pin connection	Jumper Wire color	Left Line Sensor pin connection
P0	3-Wire Jumper Cable - Yellow	OUT
3.3V	3-Wire Jumper Cable - Orange	VCC
GND	3-Wire Jumper Cable - Red	GND

Middle Line Sensor Connections:

SparkFun moto:bit pin connection	Jumper Wire color	Middle Line Sensor pin connection
P1	3-Wire Jumper Cable - Yellow	OUT
3.3V	3-Wire Jumper Cable - Orange	VCC
GND	3-Wire Jumper Cable - Red	GND

Right Line Sensor Connections:

SparkFun moto:bit pin connection	Jumper Wire color	Right Line Sensor pin connection
P2	3-Wire Jumper Cable - Yellow	OUT
3.3V	3-Wire Jumper Cable - Orange	VCC
GND	3-Wire Jumper Cable - Red	GND

Left Motor Connections:

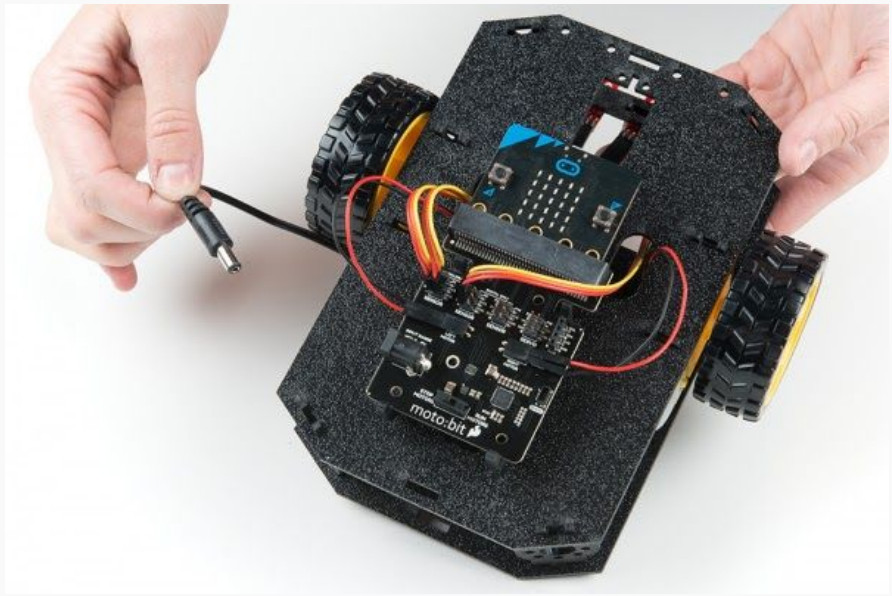
SparkFun moto:bit pin connection	Left Motor Jumper Wires	Motor Connection
LEFT MOTOR - RED	Soldered on Motor Jumper Wire - RED	(there's nothing in this column because the motor jumper wires came soldered onto the motor, so you didn't have to worry about where they were attached)
LEFT MOTOR - BLACK	Soldered on Motor Jumper Wire - BLACK	

Right Motor Connections:

moto:bit pin connection	Right Motor Jumper Wires	
RIGHT MOTOR - RED	Soldered on Motor Jumper Wire - BLACK	(there's nothing in this column because the motor jumper wires came soldered onto the motor, so you didn't have to worry about where they were attached)
RIGHT MOTOR - BLACK	Soldered on Motor Jumper Wire - RED	

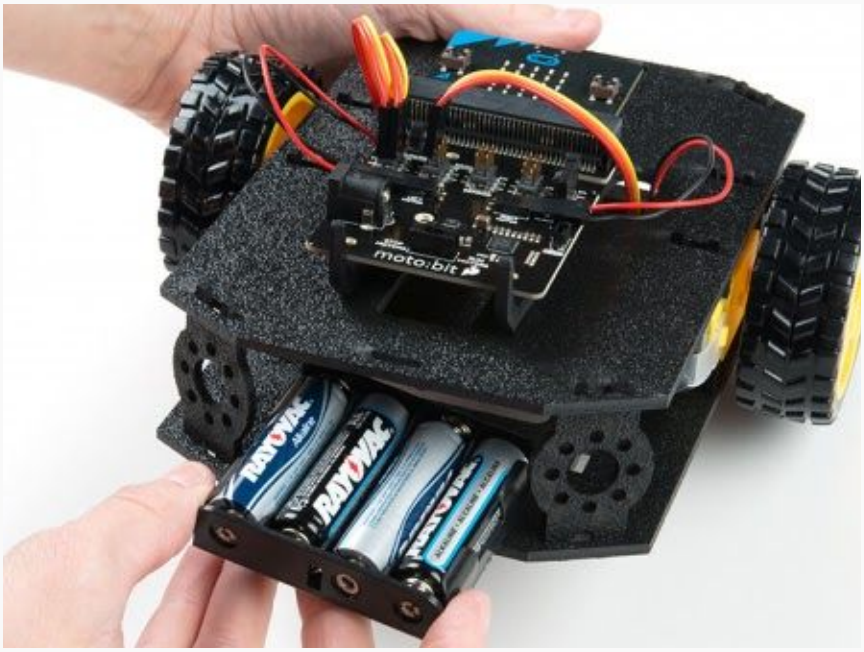
Changing the Battery

1. When you find that you need to replace the batteries in the micro:bot, first unplug the battery pack from the moto:bit.



2. Turn the micro:bot over and push on the Battery Holder through the hole in the Bottom Chassis. This will cause the Battery Pack Clip to unsnap from the Bottom Chassis.

3. Slide the Battery Pack and Clip out from the back of the micro:bot.



4. Change the batteries, then follow the steps in the "Attach Battery Pack" section above to put the Battery Pack back in the micro:bot.

Troubleshooting tips

We'll get into troubleshooting more complex issues in the next activities. For now we're focusing on five main potential sources of error. If:

Your robot won't move	
Make sure you have power! You should see a red blinking light on the moto:bit where it says "STAT" (status). If not, then →	1. Check the STOP/RUN MOTORS switch -- should be set to RUN 2. Make sure the barrel jack is completely plugged in. 3. Try new batteries
If you have power →	Make sure your motors are hooked up correctly in the motor ports
The Moto:bit tools don't show up in MakeCode →	Try installing them again from the add package option in MakeCode
You don't see the Micro:Bit on your computer when you go to transfer your code →	1. Make sure that USB cable is inserted all the way into your micro:bit. 2. Try unplugging the USB cable and plugging it back in.
Your robot is spinning in a circle or running in reverse →	Follow the steps under Fixing Motor Issues
Your robot isn't driving in a straight line →	Look at the lines in the code where you tell each motor what to do: **`move ____ motor _____ at _ %`** ● which motor you want to control (**`RIGHT`** or **`LEFT`**) ● its direction (**`FORWARD`** or **`REVERSE`**) ● the throttle / power percentage you would like the motor to spin at (**`0-100%`**) By adjusting the throttle / power percentage of the two motors relative to each other you can make one "stronger" than the other. So, for instance, if your robot lists to the left you can turn your right motor "down" to 90% power and leave your left motor at 100%. Play with the amounts to see how much correction your motors need.

License & credits

Author(s)	Gina Likins, borrowed heavily from an activity written by Derek Runberg
Source	SparkFun micro:bot guide https://learn.sparkfun.com/tutorials/microbot-kit-experiment-guide/
License	

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation. No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Code

Unless otherwise specified, all Code for the Co.Lab Robot Kit project is available under the terms of the MIT license.

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.