Using Ringo's Motors

Ringo is fitted with a pair of small motors. These allow him to move around. The ends of the motors reach down and touch the surface he is sitting on. We can make him move forward, backward, or turn depending on how we control these motors.

Controlling the motors from your code is really easy. Let's jump right into an example then we'll talk about it.

```
void loop(){
  Motors(LEFT, RIGHT);      //this function makes the motors go!
}
```

The `Motors();` function accepts two numbers as arguments: one for left motor speed, and one for right motor speed. The values can range from -255 to positive 255 for each. When this line of code is executed, Ringo's motors will immediately start running at the commanded speed.

Let's see how this works with an example on the next page, but first, let's consider a few important notes.

Ringo is not an all-terrain robot, so he does require a smooth surface to move properly. Desktops, counter tops, and smooth flooring are ideal surfaces.

Be careful Ringo doesn't jump off your table. He does have sensors to see the edges of tables, but he won't respond to them unless your code specifically tells him to do so. We'll get to that eventually, but these simple examples do not automatically look for edges.

If Ringo's motors are forced to stop, they will not be damaged due to over-current, but it's still good practice to allow the motors to spin freely without forcing them to stop.

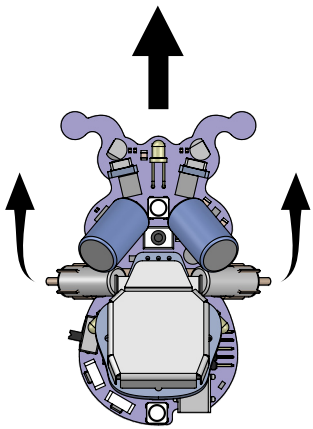
Using Ringo's Motors

Forward Movement

If we can make both motors turn in the forward direction, then Ringo should move forward. Let's try this example and see what happens:

```
void loop(){
  Motors(200, 200);         // set left and right motors to 200 speed
  delay(1000);              // wait 1 second
  Motors(0, 0);             // make both motors stop
  delay(3000);              // wait 3 seconds
}
```

In this example, we use the command `Motors(200,200);` to set the left and right motors to a speed of 200. This makes Ringo start driving forward. Then we tell his brain to wait one second with `delay(1000);` so he will drive forward for one second. Then we set both of Ringo's motors to speed 0 with the command `Motors(0, 0);` which makes both motors stop moving. Finally, the command `delay(3000);` makes Ringo wait for three more seconds. After this final line of code, the loop will repeat again from the start.



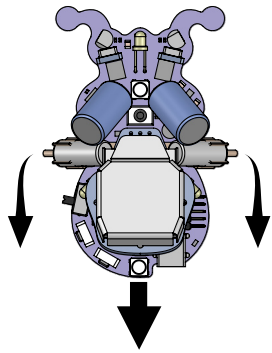
Using Ringo's Motors

Backward Movement

Now lets make Ringo move backward. Backward movement works the same way as forward movement, you just give Ringo negative movement numbers. Try this example:

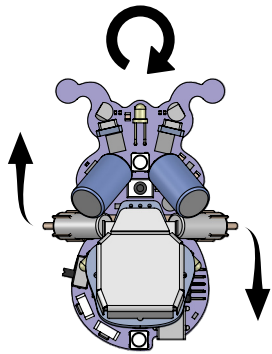
```
void loop(){
  Motors(-200, -200);       // set left and right motors to NEGATIVE 200 speed
  delay(1000);              // wait 1 second
  Motors(0, 0);             // make both motors stop
  delay(3000);              // wait 3 seconds
}
```

This example is exactly the same as the previous example, except that we've told Ringo to move the motors in the negative direction by placing minus signs in front of the speed numbers. The command `Motors(-200,-200);` does this for us.



Turning Movement

Have a guess at how we could make Ringo turn? We can set one motor to move forward and the other motor to move backward. Try using this command to set motor speed: `Motors(200, -200);` What happens? This is causing the left motor to move at speed 200 in the forward direction, and the right motor to move at speed 200 in the backward direction (the minus sign makes the motor move backward).

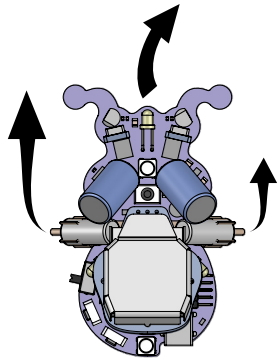


What if we reverse these numbers? Try using this command to set motor speed: `Motors(-200, 200);` We now set the left motor to negative speed and the right to positive speed. This should cause Ringo to spin in the opposite direction.

Using Ringo's Motors

Arc Movement

If you simply run both motors in opposite directions, we see Ringo turns around on his own footprint, which makes for a very agile little bug for sure. But what happens if you run both motors in the same direction, but at slightly different speeds?



Let's try running the motors like this: `Motors(200, 100);` The left motor is now running about twice the speed of the right motor. Ringo still moves forward, but he is turning to the right.

Spend a few minutes playing with the numbers in the `Motors();` function. You'll notice you can change the arc depending on how you set the numbers.

"Straight" movement?

Lets go back to where we started in this discussion and look at forward movement a bit more closely. Let's try the example code below. Ringo is going to drive much further this time, so you may want to test this example on a floor. If you're doing this on a table, be ready to catch him before he jumps off the edge.

```
void loop(){
  Motors(150, 150);        // make both motors run
  delay(3000);             // wait 3 seconds
  Motors(0, 0);            // make both motors stop
  delay(5000);             // wait 5 seconds
}
```

Did Ringo drive in a perfectly straight line? We set both motors to forward speed 150, so both motors should be turning at the same speed, right? You would think Ringo would have been driving in a perfectly straight line but I'm guessing he didn't (though you may have gotten lucky this time and perhaps he does travel perfectly straight, but not likely).

Using Ringo's Motors

Let Ringo drive this "straight" line a few times. You'll probably notice he tends to veer off one direction or another. Study the Arc Movement example above and see if you can adjust the numbers you use in the `Motors();` function to make him go a bit more straight. Hint, if Ringo veers to the right, increase the right motor number a bit, and if he veers off the left, increase the left motor number a bit.

Mechanical Systems Aren't Perfect

Let's talk more about the previous experiment. Do you have any idea why Ringo didn't go straight even when the same motor speed numbers were used? The reason is that mechanical systems are never made perfectly. Each motor will run at a slightly different speed even when driven with the same power source as one will be slightly more efficient than the other. There are lots of other variables at play also - the surface may be more bumpy on one side, one motor tip may be getting better grip on the surface, etc. All mechanical systems have some variation. It's possible to minimize the variation (by using very expensive and super accurate motors for example), but some variation will always exist.

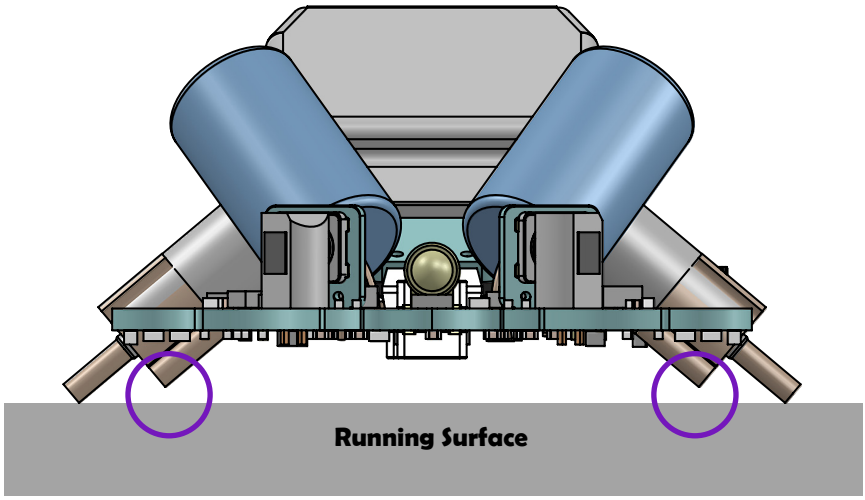
It isn't very practical to constantly guess and adjust a robot's speed settings to make it go in a straight line as you experienced in the last example. For this reason, most robots require some way of sensing how they are moving. Wouldn't it be smart if Ringo could somehow know that he was starting to veer to the right and automatically adjust the speed of his motors to compensate and resume a straight path? Well lucky for Ringo (and you!), he has two sensors that can be used for this purpose. One is called a Gyroscope, which basically allows Ringo to know when he is turning. The second one is called an Accelerometer, which basically allows Ringo to know how fast he is moving and how far he has traveled. Using these awesome sensors is beyond the scope of this lesson on basic motor operation, but check out the Navigation lesson later in this guide for the full run down.

Using Ringo's Motors

It no workey. (Sad Ringo).

The motors are the only real "mechanical" part of Ringo. It is important that they are oriented correctly to the running surface to operate properly. They may occasionally become misaligned in the motor clip, become bent, or get gummed up with hair or other yuk that Ringo may have driven through. Here is a quick guide to inspecting Ringo's motors and fixing any issues you may encounter.

1) Mis-Aligned Motor Clip: Make sure Ringo's motor clip is not dragging on the running surface. Look straight on to Ringo while he is on a surface. Look at the areas with the purple circles. There should be a gap of about 1mm between the running surface and the lowest portion of Ringo's motor clip.



If you find the motor clip is dragging, it can be corrected by making two adjustments.

a) Make sure Ringo's motors are flush with the end of the motor clip. If the motors have been pushed further inward into the clip, the shaft is effectively shortened. Re-set motors to flush with the clip.

b) The clip is spring steel and should be very difficult to deform, but you can carefully bend the ends of the motor clip downward. This causes a greater angle with the surface and thus creates more space between the clip and the surface.

Using Ringo's Motors

2) Bent Motor Shaft: This should not be a common problem, but it is possible. If Ringo takes a fall onto a hard surface and happens to land right on one of his motor shafts, it is possible the shaft could become bent. In all our testing, our Ringos have taken many hard falls onto a concrete shop floor from 3 to 4 feet heights. In all these drops, we managed to slightly bend one motor shaft which was easily fixed as follows. You'll notice it is bent because as Ringo moves, one side will "thump" along rather than running more smoothly. Correcting a bent motor shaft is possible with some care. Closely examine the shaft to determine the direction it is bent, then, using a tool of some sort, carefully reverse the bend. This will likely return Ringo to service. If the bend is so extreme that it cannot be reversed, motor replacement may be required. Contact us to order replacement motors.

3) Gunk on Motor Shaft: Ringo is very good at picking up debris in his motor shafts - especially hair from your pets when he is run on the floor. Usually this is easy to remove. Using tweezers, carefully pluck the hair off the shaft by rolling the shaft with your fingers. A bit of hair build up isn't a major problem, but if enough is allowed to accumulate, it can effect the performance of the bearing inside the motor where the shaft exits.

WARNING! If you do feel the need to remove a motor from the clip (this shouldn't be required, but we know some of you will try to do it anyway) - the best way to remove the motor is to slide it completely through the clip outward. It is tempting to rock the top end of the motor (where the wires come out) upward until it prys open the clip then rotate it outward. THIS IS A BAD IDEA THOUGH. As the motor is rotated out of the clip, all the force is placed on the motor shaft as the motor shaft hits the bottom of the clip. This force (if pryed all the way out of the clip) is enough to bend the motor shaft. Instead, slide it through the clip until it pops out the end. Be careful you don't break the wires when doing this. To re-install the motor, align it over the clip so the end is flush, then use finger pressure to snap it into the clip from the top through the open "jaws" of the clip.