

Circuit Playground Express

*Motors – Continuous Servo*

This lesson introduces the first of the motors that we will be exploring. There are three kinds of motor that we’ll be learning about: Continuous servos, standard servos, and DC motors. Each motor will have its own lesson plan that shows how to use the motor and also introduces new programming concepts.

**DC motors** – These are the simplest type of motor, the least expensive, and the most powerful. The drawback is that they are very complicated to incorporate into a program; controlling them with code requires a component called an H-bridge that is hard to understand and complicated to wire up correctly.

**Continuous servos** – These work very similarly to DC motors, but they have a slightly higher cost and tend to be less powerful. The advantage that they bring is much greater simplicity: they are very easy to wire up, and with a single MakeCode block you can control their direction and the speed at which they rotate.

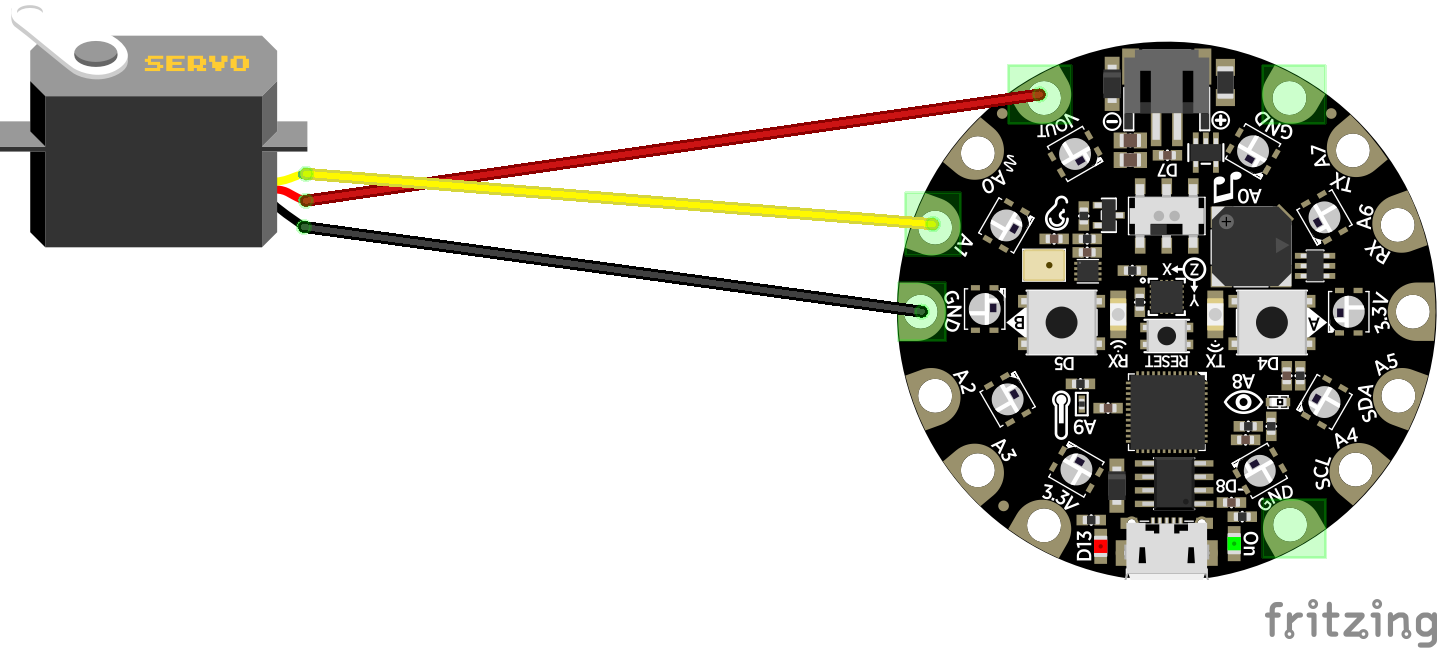
**Standard (positional) servos** – These are similar to continuous servos in terms of power, cost, and simplicity. However, they are different in one key way: Whereas DC motors and continuous servos spin continuously, standard servos can be programmed to rotate to a specific angle. They can only rotate back and forth within a given range of motion, but for applications where it’s important that a component rotate to a specific angle, they are invaluable.

In summary: If you need a motor that spins around and around, go with a continuous servo or DC motor. For simplicity, the continuous servo is better, but the DC motor will provide a bit more power and cost less. If you need a motor that stops at a precise angle, use a standard servo.

A Basic Servo Program

Wire up your continuous servo as below. A servo has three wires:

* A power supply wire, which should be wired to VOUT. This wire provides the power that spins the motor.
* A control wire, which can be wired to any of the A0-A7 pads. This wire carries the signal that tells the servo which direction and how fast to spin, and is what we will use to program the servo.
* A ground wire, which goes to one of the GND pads provides the current in the circuit a destination to flow to.



Now let’s program our servo to actually spin.

There are blocks that make working with a servo very easy, but they are in an extension that we will have to install. Under “Advanced” in the block catalog, find “Extensions” and then click “servo.” Once you’ve done this, you should see a new “Servos” category in your block catalog.

With the help of one of the blocks from the new “Servos” category, build the following program:



Download the program to your Circuit Playground Express and see if it works. It should spin the servo at maximum speed in one direction for two seconds, then reverse for two seconds, and continue to do this in a loop.

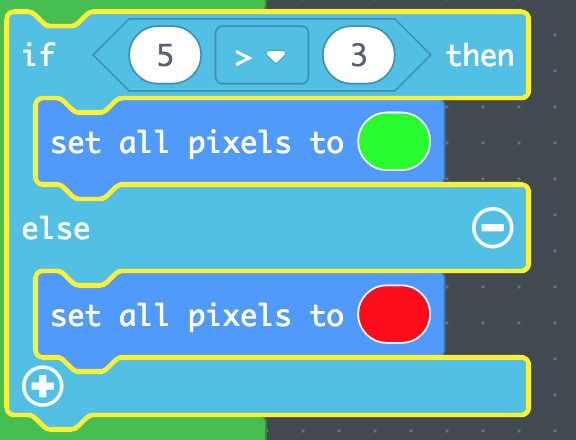
Try playing with the speeds. How fast is 50%? 10%? It’s good to experiment with a new component and get a sense of what its range of capabilities are.

Controlling the Direction with the Switch

Let’s make our servo respond to the direction the Circuit Playground Express’s switch is facing. Look under “Inputs” for the “switch right” block, and under “Logic” for the “if/else” block.



How does an if/else block work? It checks the *condition* it is given, and if the condition is true then the code under “if” runs – if it is not true, then the code under “else” runs.



This code will run if the *conditional* is *not* true

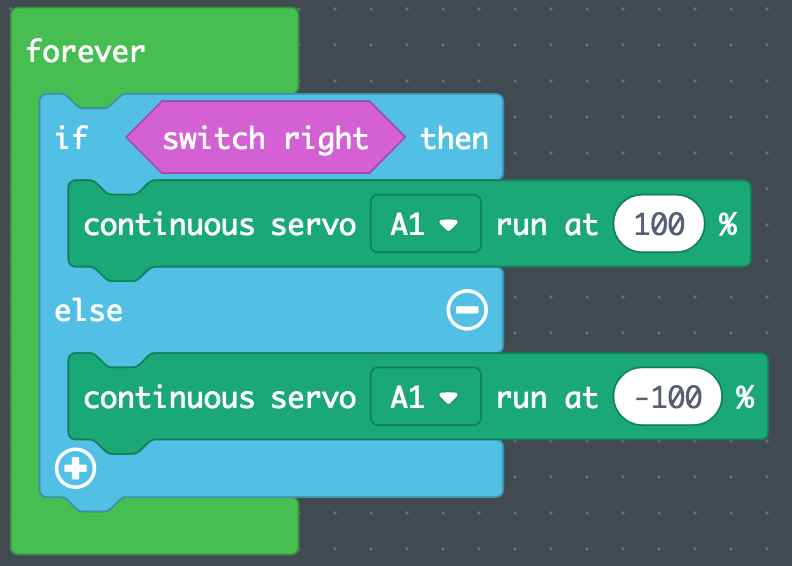
This code will run if the *conditional* is true

The *conditional*



Note that the *conditional* part of an if/else block is shaped like an elongated rhombus (or “pointy-sided thingy” for the non-geometry nerds) – any block that has that shape can be dropped in. These are mostly found under “Logic” and “Input.”

In our case, we want to use the block “switch right”, which is a conditional that is true if the switch on the Circuit Playground Express is pushed to the right, and is false if it is pushed to the left. Can you see how you could use this to create an if/else block that will spin your continuous servo in the direction indicated by the switch?

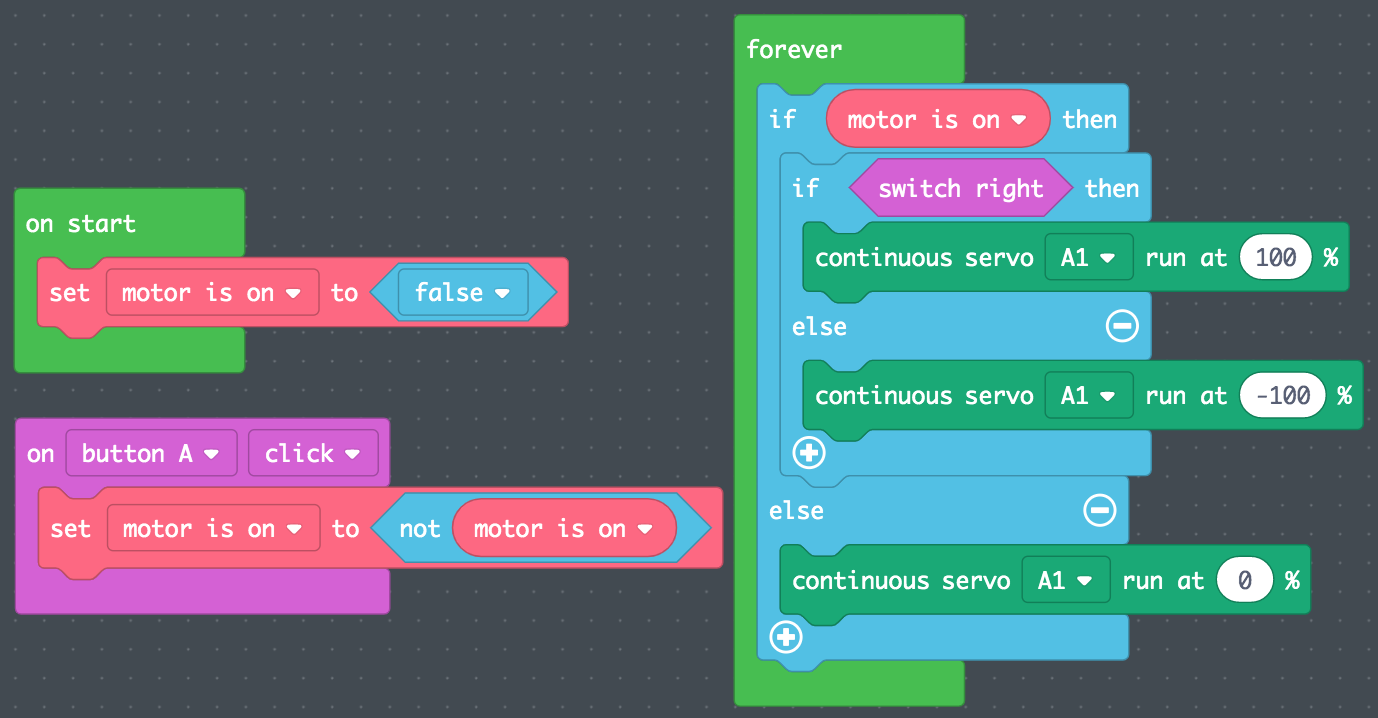


Introducing Variables – Adding an On/Off Button

A very important concept in programming are *variables*, which are containers for values. The current version of our program makes the motor just run and run without ever stopping, but we are going to use a variable to make it possible for a button to control whether the motor should run or not.

To create a variable, go to “Variables” in the block catalog, and click “Make a Variable.” You can give a variable any name you like, but it’s important to pick a name that makes sense to you so that your code is easier to understand. The variable we’ll be creating will contain a value that indicates whether the motor is on, so I am going to name mine “motor is on”.

Now assemble the following program:



*Tip: If you are looking for a block, it might help to note that they are color-coded! Blue blocks are always under “Logic,” purple blocks are always under “Inputs,” and so on.*

How do these three blocks work?

On start: This block runs once as soon as the Circuit Playground Express is turned on. We have to give our variable an initial value, “false,” meaning that we don’t want the motor to run initially.

On button A click: Whenever the A button is clicked, we are going to switch “motor is on” between the value “true” and the value “false.” That’s what the “not” block does – <not true> equals false, and <not false> equals true.

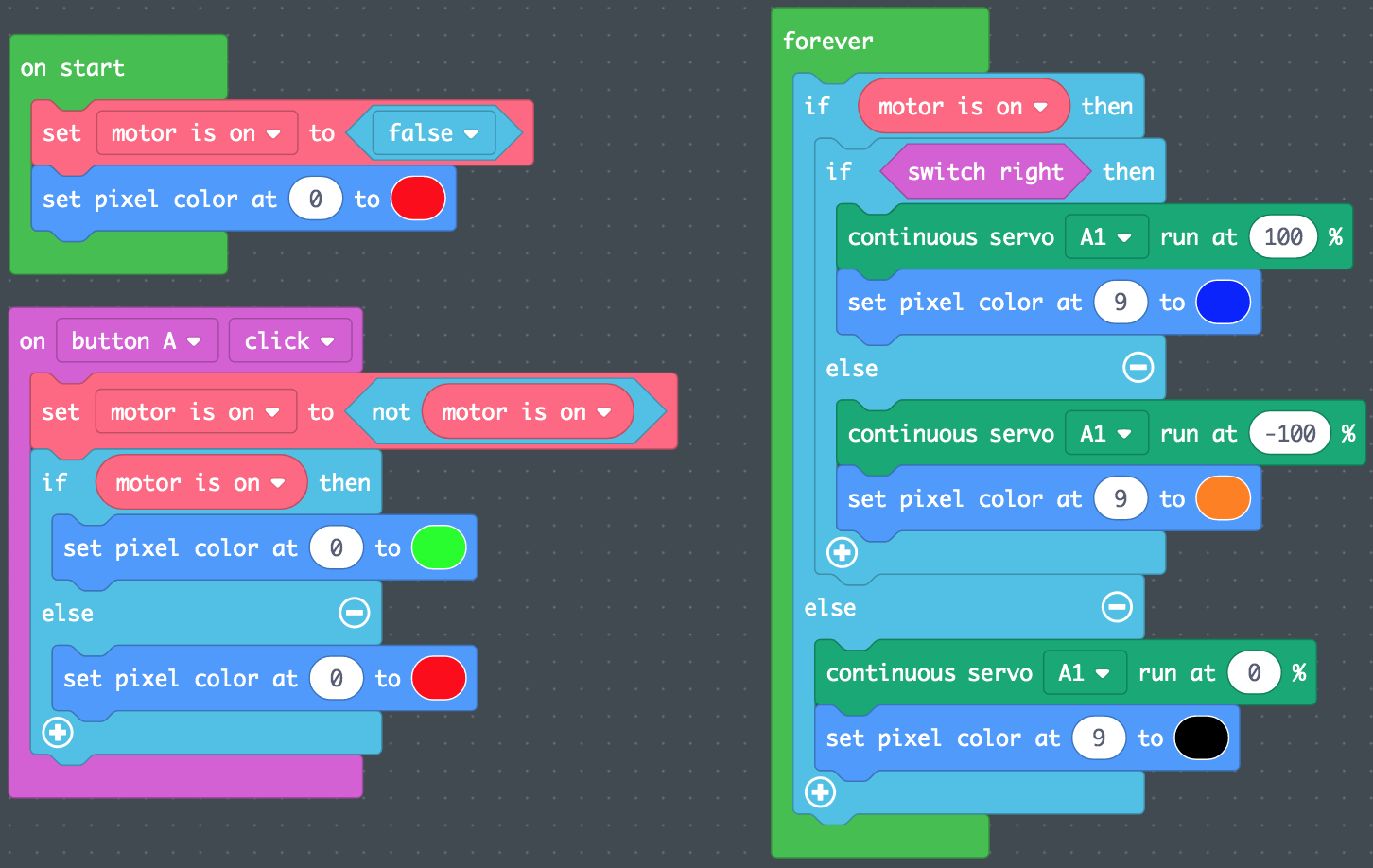
Forever: This is the same as our previous forever loop, except that now we also check our “motor is on” variable; if it is true, then we spin our motor the direction indicated by the switch, and otherwise we stop the servo altogether. *Note: We use a “run at 0%” block to stop the servo. There is a “stop servo” block in the catalog, but it can cause errors when attempting to re-start the servo.*

Upload your program and give it a try! If everything works correctly, you should be able to turn your motor on and off with the A button, and change the direction it spins with the switch.

Troubleshooting Using LED Indicators

Chances are, by now you’ve had at least one bug in your programs. That’s normal! These systems are complicated, and getting all the parts right the first time is nearly impossible.

One strategy that can be very helpful in diagnosing a Circuit Playground Express project that isn’t working is adding indicator LEDs to your program to help you see what your code is doing. For example, this version of our servo program, includes LED indicators so that we can see the state of the “motor is on” variable and also the signal we are sending to the servo:



Give this program a try. Can you see how indicating the state of your program by using LEDs can make it easier to visualize what your code is doing? This trick can be very helpful if your project is not working but you’re not sure whether the problem is in the hardware or in your code. If you can verify by looking at your LEDs that the code is doing what it ought to be, then you can be more confident that the issue is not a programming bug. And it can make it much easier to identify a programming error – if you push the A button but pixel 0 doesn’t change color, you’ll know to look for a bug in the “on Button A click” block.