

Circuit Playground Express

*Positional Servo and Proximity Detector*

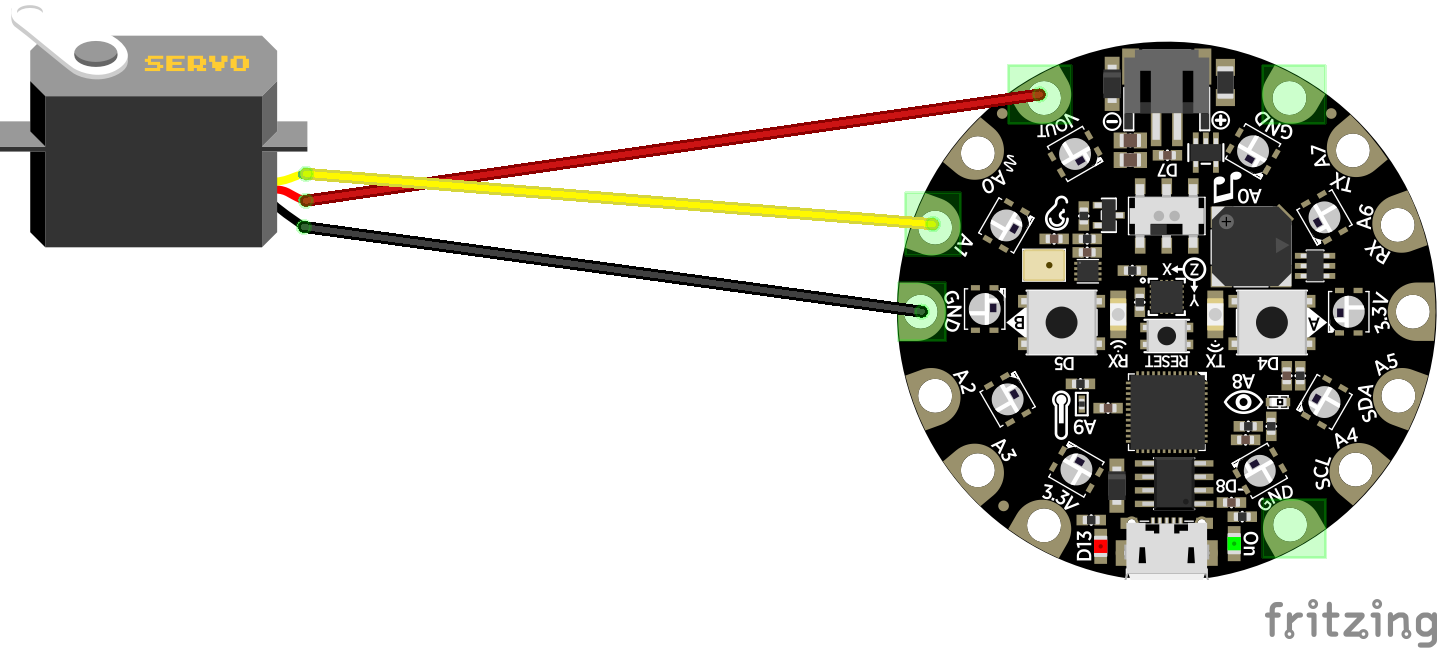
This lesson introduces the positional servo, as well as the proximity detector sensor. Recall from our previous lesson that a positional servo is a motor that cannot spin around and around but can be programmed to rotate to a specified angle. These motors are very useful for tasks like opening and closing doors or pointing a sensor in a given direction.

*Note: Different positional servos have different ranges of motion. This document was written using a Micro Servo A0090, which has a range of motion of 120°. Other servos might have a greater or lesser range of motion than that, but will otherwise work the same.*

A Basic Directional Servo Program

A positional servo is wired up just the same as a continuous servo:

* 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 angle to rotate to.
* A ground wire, which goes to one of the GND pads provides the current in the circuit a destination to flow to.



Just to make sure our servo is working as expected, let’s write a simple program like we did in the previous lesson, to spin the servo between two positions.

A green sign with white text

Description automatically generated

A Basic Proximity Sensor Program

An ultrasonic proximity sensor uses very high-frequency sound to measure the distance to an object; it emits several pulses of sound and then listens for an echo and reports back the amount of time that the sound took to reflect back at it. Since we know how fast sound travels through air, with a bit of math you can work out the distance to the object that the sound bounced off of.

A cup of coffee

Description automatically generated

A proximity sensor has four pins:

VCC – This pin provides power, and should be wired to the VOUT pad

GND – This pin provides a path to ground, and should be wired to a GND pad

Trig – The trigger pin is used to activate the sensor. The sensor will be triggered whenever the voltage on this pin is set to “high” for at least 10ms, then set back to “low.”

Echo – After the trigger pin has been activated, this pin carries the result back to the Circuit Playground Express. The number it reports is the number of microseconds (millionths of a second) that passed between the sound being emitted and the echo returning.

Let’s wire up our proximity sensor using a breadboard.

A close up of a device

Description automatically generated

Now add the following code to turn the Circuit Playground Express’s LEDs into a display of the distance to the object the proximity sensor is detecting:

A screenshot of a cell phone

Description automatically generated

This is a bit more complex than the programs we’ve written up to this point, so let’s examine what each section means.

A picture containing road

Description automatically generated

Pin A1 is connected to the Trig (trigger) pin on the proximity sensor. Setting the voltage on that pin to high for 10ms, then setting it back to low, triggers the sensor.

A red and white sign

Description automatically generated

This sets a variable called “echo\_duration” to the number of microseconds between the sensor emitting its ultrasonic pulses and those pulses returning to its microphone. The “pulse in” item means “get me the number of microseconds that the voltage coming into pin A2 is set to high,” which is how the ultrasonic sensor communicates the echo duration.

A screenshot of a cell phone

Description automatically generated

“graph” is a neat block that allows you to display numeric data in the form of LEDs turning on and off on the Circuit Playground Express. It’s written to be pretty smart; it will automatically adapt to whatever numbers you give it, so all lights turned on basically represents “close to the largest number the graph block has received since the program started”, and one light turned on represent “close to the smallest number.”

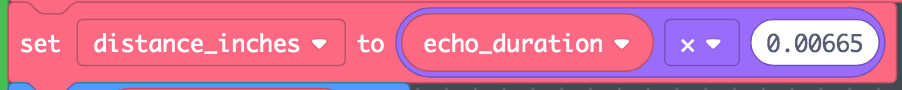


This block is not strictly necessary, but the LEDs will flicker a great deal if you don’t introduce a bit of delay between updates.

Triggering a Servo Using the Proximity Sensor

Of course, just displaying the distance to an object is neat, but it isn’t very practical. If we are using a proximity detector, we probably want to trigger some behavior when an object gets within some distance. How might we go about that?

For starters, lets convert the echo duration into inches to make it a bit more understandable. Given the speed of sound, we can work out that every microsecond represents 0.00665 inches between the ultrasonic sensor and the object its sound is reflecting from.



Remember what we learned about conditionals (“if” blocks and “if/else” blocks). Try making a program that turns an LED on if an object is within 10 inches of the Circuit Playgroun Express, and turns it off otherwise. What might that look like? (Give it a shot, then turn to the next page for one solution in case you are struggling to work it out)

A screen shot of a smart phone

Description automatically generated

Now let’s wire up a servo in addition to the proximity sensor, so that we can trigger the servo to rotate when the proximity sensor detects a nearby objects. Note that we are now using the + (power) and – (ground) busses on the breadboard. This makes it possible for multiple components to easily connect to the same power supply and ground pads.

A close up of a device

Description automatically generated

Now we can update our previous program to spin a servo instead of blinking an LED.

A screen shot of a smart phone

Description automatically generated

Note that the “echo” pin of the ultrasonic sensor has been moved to A7. This is to free up A2 for the servo – for technical reasons, only A1 and A2 are capable of sending the signal required to control a servo.