Using a servo with the micro:bit

Modified on: Wed, 5 May, 2021 at 1:21 PM

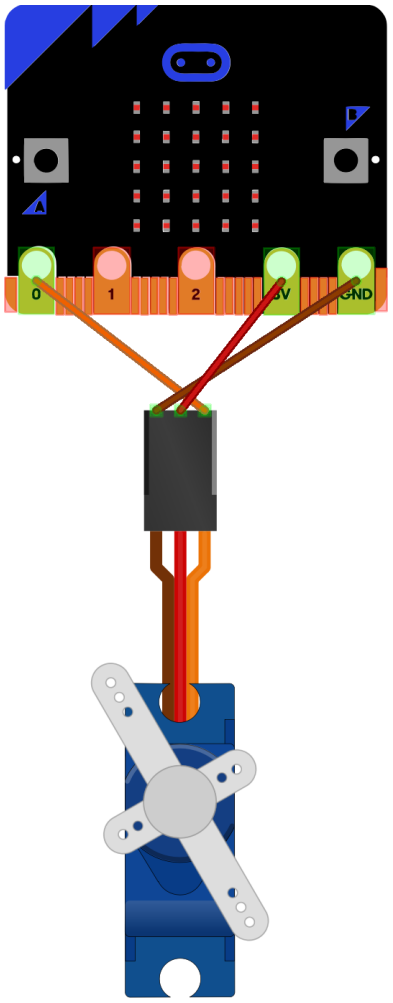
Overview

This article explains how to connect a servo motor to the micro:bit and how to code it in the micro:bit editors. It also provides some troubleshooting and further information on use.

Contents

Connecting a servo

It's easy to connect up a servo to the micro:bit either using crocodile/alligator leads or a breadboard. A micro-servo such as the **SG90** (https://components101.com/servo-motor-basics-pinout-datasheet) or Tower hobby servo (either 180-degree rotation or 360 degrees) can be connected from Pin0, 3V and GND and controlled by sending the signal on Pin0. Usually the wiring colouring is Orange = Signal, Red = 3V, Brown = Ground(GND)



Connecting a servo motor to the micro:bit

Whilst these micro-servos can work with the micro:bit, the specified operating voltage for most servo motors is around +5V and that the micro:bit can only supply a **small amount of power to connected circuits** (https://tech.microbit.org/hardware/#power-supply) (https://tech.microbit.org/hardware/#power-supply)**(3V** (https://tech.microbit.org/hardware/#power-supply) **and 90mA V1 and 190mA V2max).** (https://tech.microbit.org/hardware/#power-supply) Trying to draw more power than the micro:bit can safely supply, could lead to damaging the device.

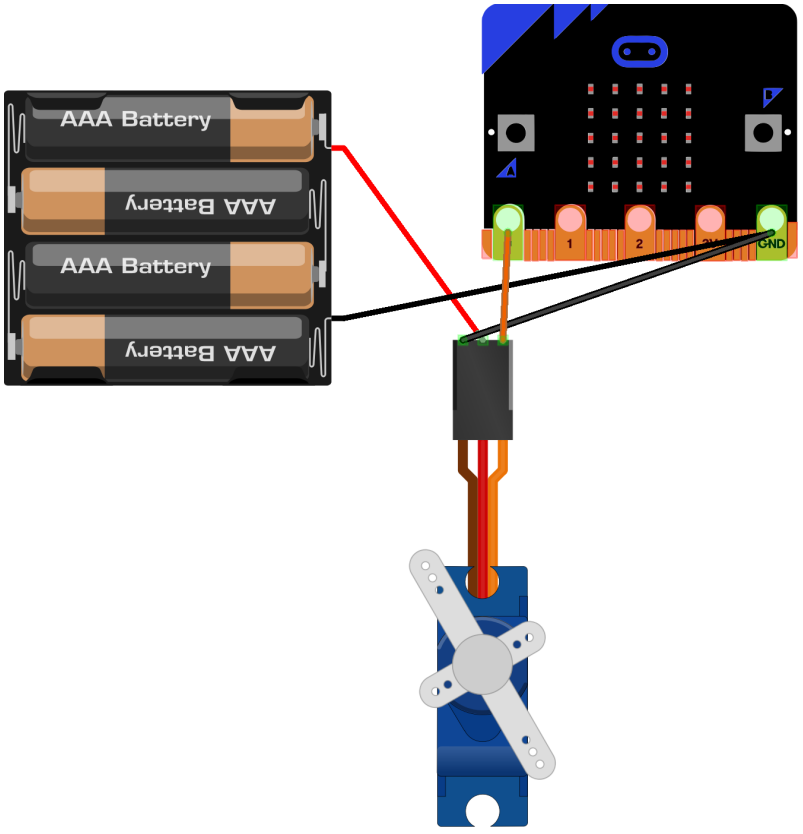
For micro:bit V1, the most reliable way to use this type of servo is to power the micro:bit via a battery pack and to use fresh batteries, as the battery voltage drops the servo will become less reliable.

Connecting an external power supply to a servo

The optimal method for connecting a servo is to use a separate battery pack to power the servo and use the micro:bit to control it. This way you are only connecting Pin0 and GND from the micro:bit to the servo (we still need to use GND to share a common ground with other parts of the circuit).

The external battery pack supplies a higher voltage than the micro:bit. Do not connect the positive (+/red) wire from an external battery pack to the micro:bit as you will damage it.

Additional battery packs often come as either 4.5V (3 batteries) or 6V (4 batteries). The micro:bit will supply 0V or 3V on the PWM pin0, and this has to be above the digital input pin threshold of the servo (this will be defined in the servo datasheet and often this is 0.7*VCC). If the voltage supplied to the servo is too high, half of that voltage becomes the pin threshold and the 3V from the micro:bit might not be enough to direct the servo.



With the extra connections, you may find it easier to use a **breadboard** (https://components101.com/misc/breadboard-connections-uses-guide) and a micro:bit edge connector breakout to make building the circuit easier.

Programming a servo

Servo motors determine their position by the ratio of on-time and off-time in an (approximately) 20 millisecond (ms) pulse. The micro:bit makes use of **Pulse Width Modulation (PWM)** (https://learn.sparkfun.com/tutorials/pulse-width-modulation) as a way to simulate an analogue output on a digital pin. It sends a series of high speed on/off pulses to the servo which sets its target position. eg 0, -90, 180 degrees, or in the case of a continuous rotation servo, the speed and direction of rotation.

Makecode

To program the micro:bit to control the servo, we will need to send a signal to it on Pin0. **MakeCode has a handy reference for servos that describes the use of the Servo Write Pin block.** (https://makecode.microbit.org/reference/pins/servo-write-pin) Note the difference in use for 180-degree rotation and 360 degrees / continuous rotation servos.

Python

In MicroPython, we need to set our initial PWM period for the pin and then write an analogue value to it

```
from microbit import *
# Servo control:
# 50 = ~1 millisecond pulse all right
# 75 = ~1.5 millisecond pulse center
# 100 = ~2.0 millisecond pulse all left
pin0.set_analog_period(20)

while True:
        pin0.write_analog(75)
        sleep(1000)
        pin0.write_analog(50)
        sleep(1000)
        pin0.write_analog(100)
        sleep(1000)
```

There is a **video example of driving a servo in MicroPython with a useful demonstration of PWM** (https://www.youtube.com/watch?v=k4iM5fXt54k). There is also a third party **micro:bit servo class** (https://github.com/microbit-playground/microbit-servo-class) that you can import using the micro:bit filesystem available in MicroPython.

There are some mixed standards as to what pulse width causes what specific servo position (or servo speed and direction). This data can be found by searching for the datasheet for the model number of the servo eg SG90.

Troubleshooting using a servo with the micro:bit

If the servo is juddering or is not moving at all. Try these troubleshooting tips:

- Power the servo(s) externally, not from the micro:bit. The micro:bit can only provide 3V and most servos operate at around 5V.
- On micro:bit V2, battery power supply goes through an on-board voltage regulator. This is not the case on micro:bit V1, where the servo draws power directly from the battery.
- Use fresh batteries
- Tap the servo arms to start it moving. A servo requires more power to get it started than to keep it moving. The power consumption also depends on what is connected to the servo. A heavier object will require more power.
- Check the type of servo you have. For 180 degree servos, you can control target position, for continuous rotation servos you can control direction and speed of rotation.
- If your project uses multiple servos, consider a dedicated servo motor breakout board/accessory. The amount of **PWM pins available** (https://tech.microbit.org/hardware/edgeconnector/#edge-connector-pins) may limit the amount of servos you can control.

Further Information

Kitronik's brief guide to servos (https://www.kitronik.co.uk/blog/servos-brief-guide/)