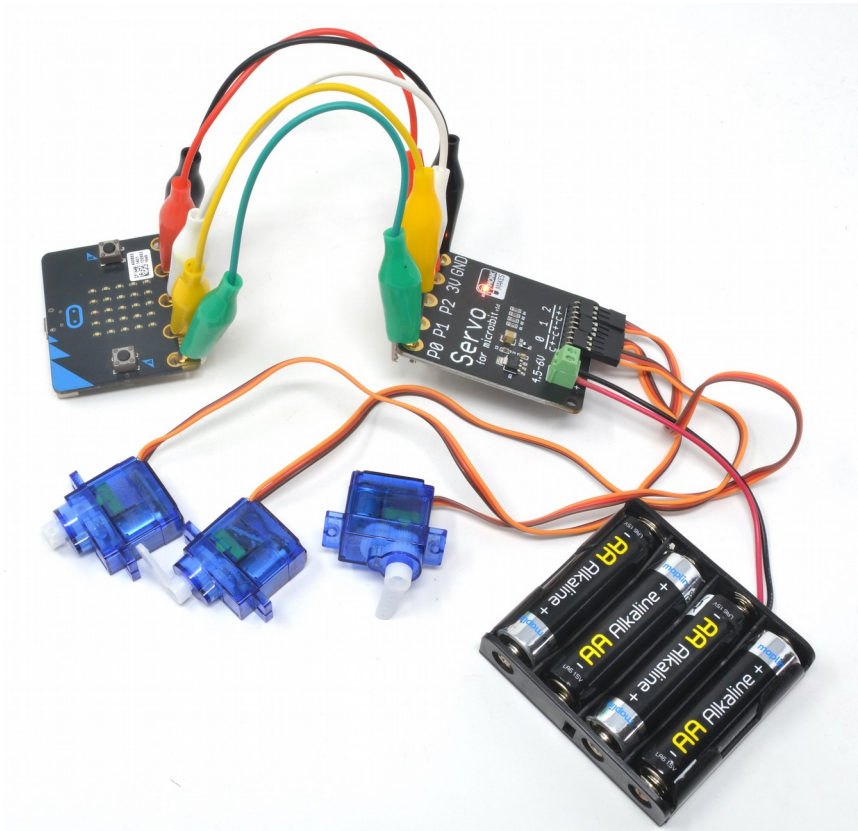


# Instructions:

## SERVO KIT

FOR MICRO:BIT

v1c







# TABLE OF CONTENTS

Parts .....	3
Connecting Everything.....	4
Programming Servos.....	8
Servomotors.....	9
Troubleshooting.....	10
Learning.....	11
MonkMakes.....	12

# PARTS

Please note that BBC micro:bit and batteries are not included in this kit.

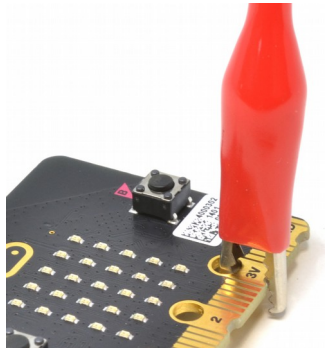
Before you do anything else, check that your kit includes the following items:

Quantity		
1	MonkMakes Servo for micro:bit	
3	9g micro-servomotors, each with a selection of servomotor arms and screw for attachment.	
1	4 x AA battery box for powering the servomotors and your micro:bit.	
1	Set of alligator clip leads (10 leads)	

# CONNECTING EVERYTHING

## About Alligator Clips

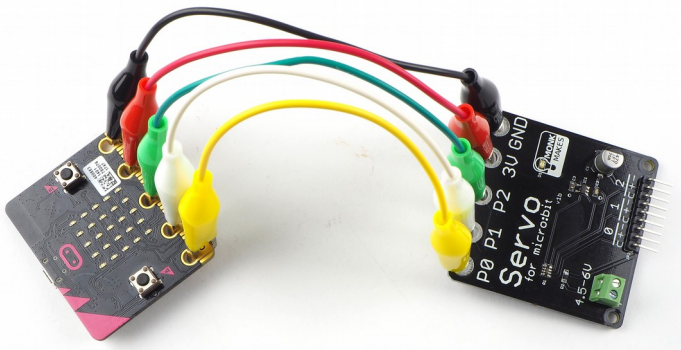
When using the alligator clips to connect your micro:bit to the MonkMakes Servo for micro:bit board, you have to be a bit careful how you connect the clips at the micro:bit end. The best way is to connect the clips vertically as shown below.



Connecting the alligator clips like this prevents any accidental connections between the large connectors with the holes in and the much smaller connectors (gold lines in the photo above)

## 1. Connect the micro:bit to the Servo for micro:bit

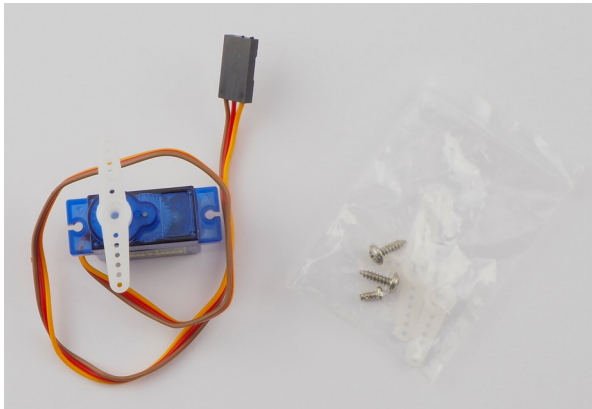
Use five of the alligator clip leads to connect your micro:bit and Servo for micro:bit together.



Its a good idea to use the red lead for 3V, black for GND and whatever colors you like for the other connections.

## 2. Prepare the Servomotors

Take the servomotors out of their bags and remove one of the servo arms from the bag of arms and screws in each servomotor's bag. Push the servo arm onto the shaft of the servomotor as shown below.



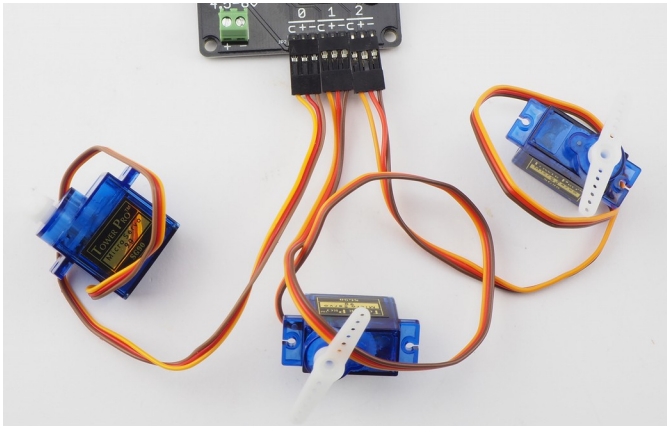
The smallest of the screws in the servomotor's bag is intended to secure the servo arm into place, but you don't need to use it, as the arm should stay in place without it.

## 3. Attach the Servomotors

Now you can attach the servomotor's leads to the header pins sticking out of the edge of the Servo for micro:bit board. Be careful, these can be sharp and scratch.

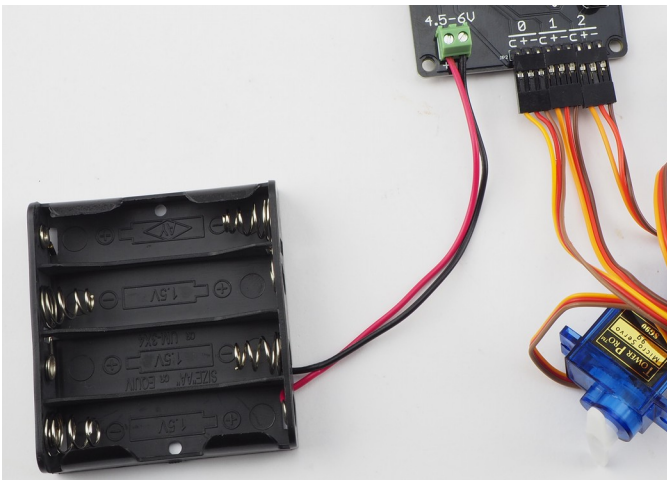
The servomotors must be connected the correct way around, with their orange control connections to the left as shown on the next page.

Make sure that the servomotor's connectors are pushed properly into place on the pins.



#### 4. Connect the Battery Box

The battery box has two leads coming out of it that need to be connected to the screw terminals on the Servo for micro:bit board.




Using a screwdriver, unscrew both of the terminals in the terminal block and then insert the end of the red lead into the screw terminal marked with a + and do up the screw of the terminal. Do the same with the black lead in the terminal marked with a -.

You can now put the batteries in the battery holder, making sure you get them the right way around to match the drawings inside the battery box. You may find that

the servomotors shudder as the last battery is put into place.

## 5. Program the micro:bit

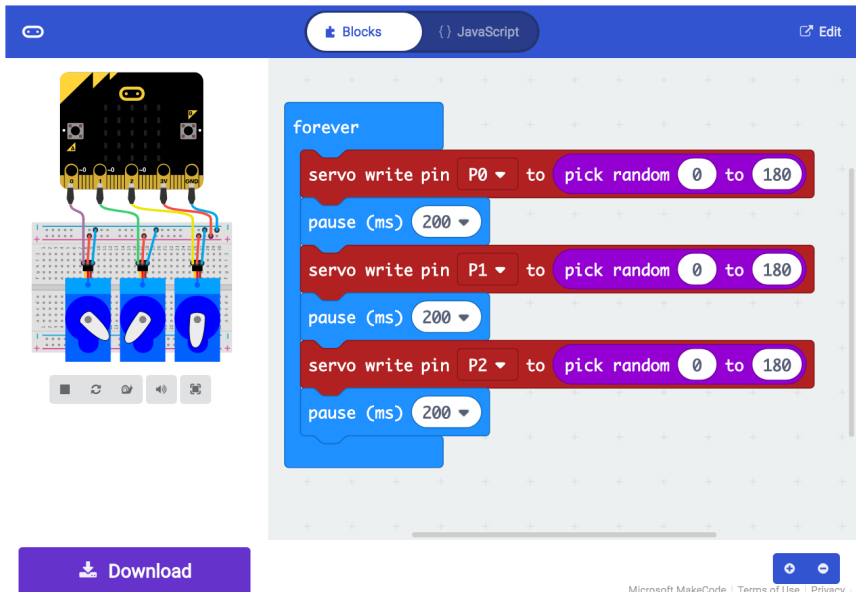
The servomotors are not going to do anything until you flash a program onto the micro:bit. So connect your micro:bit to your computer using a USB lead and then visit this webpage in your browser. [https://makecode.microbit.org/\\_86KWdwTs9VRo](https://makecode.microbit.org/_86KWdwTs9VRo)

 **User-provided content**  
The content below is provided by a user, and is not endorsed by Microsoft. If you think it's not appropriate, please [report abuse](#).

Untitled

Edit

5 months ago



Click on the *Download* link at the bottom of the window and copy the HEX file onto your micro:bit. Once flashing is complete, the servomotors should spring into action moving by random amounts.

Carry on reading to find out more about how to program your servomotors as well as information about how servomotors work.

# PROGRAMMING SERVOS

When it comes to programming your micro:bit to make the servos move, you can use either Block code, as we just have, or you can use MicroPython.

## Blocks code



The only block you need to control a servomotor is the *servo write pin* block that you will find in the *Pins* category of the Blocks editor.

The block contains a drop-down list, where you specify which servomotor you want to change the arm position of. The value of 180 is the angle of the servo arm in degrees. The angle can be set to any value between 0 and 180 degrees. Where 0 and 180 are opposite ends of the arm's position and 90 is the center position.

Once you have used the servo write block on a particular pin, the servomotor will stay in that position until you set it to a different position.

## MicroPython

Controlling servomotors using MicroPython is more complicated. There is no built-in function for setting a servo to a particular angle. The following example includes a function called *set\_servo\_angle* that does the same job as the *servo write block*.

The example below allows you to control the angle of a servo attached to pin 2, using the A and B buttons on the micro:bit. Pressing button A will decrease the servo angle by 10 degrees and pressing B will increase it by 10.

```
from microbit import *

def set_servo_angle(pin, angle):
    duty = 26 + (angle * 102) / 180
    pin.write_analog(duty)

angle = 90
set_servo_angle(pin2, angle)

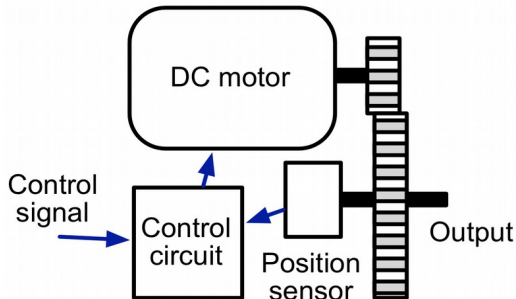
while True:
    if button_a.was_pressed() and angle >= 10:
        angle -= 10
        set_servo_angle(pin2, angle)
    if button_b.was_pressed() and angle <= 170:
        angle += 10
        set_servo_angle(pin2, angle)
```

If you want to create your own programs in MicroPython for controlling a servomotor, I suggest you start by copying the *set\_servo\_angle* function.



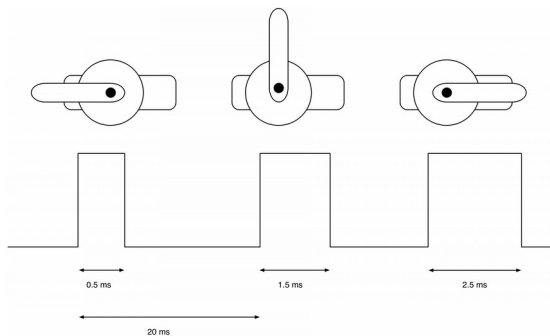
# SERVOMOTORS

Despite their small size, servomotors are surprisingly complex little machines, controlled using signal pulses. Here's a diagram of how a servo motor works.



A DC motor (normal motor that can rotate continuously) is attached to a gear box, which drives the servo arm, but is also coupled to a position sensor that provides feedback to an electronic control circuit that keeps the servo arm at the correct angle.

Control of the position uses a series of pulses arriving every 20 milliseconds.



The length of each of these pulses will determine the position of the servo arm. So a short pulse of just 0.5 milliseconds will put the arm at one end of its travel. A pulse of 1.5 milliseconds will put the arm at its center position and a maximum pulse length of 2.5 milliseconds will put it at the other end of its travel.

# TROUBLESHOOTING

**Problem:** The servomotors don't move, and the Servo for micro:bit boards orange LED is NOT lit.

**Solution:** This means that the Servo for micro:bit board is not getting power. This could be for a number of reasons:

- The batteries are depleted
- The battery box is connected the wrong way around
- One or more of the batteries are the wrong way around

**Problem:** The servomotors don't move, and the Servo for micro:bit boards orange LED is lit.

**Solution:** Power is getting to the Servo for micro:bit board, so this means that the cause is probably one of the following:

- The connections between the micro:bit and the Servo for micro:bit board are incorrect. See step 1 in *Connecting Everything*.
- The servomotors are connected the wrong way around. See step 3 in *Connecting Everything*.
- The micro:bit has not been programmed to move the servomotors. See step 5 in *Connecting Everything*.

**Problem:** When I set the servo angle to 0 or 180 the servo judders.

**Solution:** This is quite normal. Many servomotors will judder when they get to the ends of their range of angles. If you get this problem, then try reducing the range of angles that you try to set your servomotor to. Perhaps 10 to 170 rather than 0 to 180.

**Problem:** When I move the servo angle between 0 and 180 in the software, the actual angle moved by the servo is more than (or – more likely – less than) 180 degrees.

**Solution:** This is quite normal. Most servomotors do not exactly match the angle they are commanded to by their control pulses. See also the previous problem.

For other support questions, please check out the product's web page ([http://monkmakes.com/mb\\_servo\\_kit](http://monkmakes.com/mb_servo_kit)) and you can send support questions to [support@monkmakes.com](mailto:support@monkmakes.com)

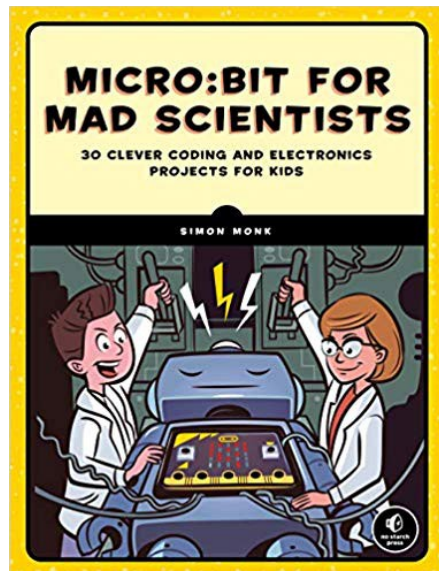
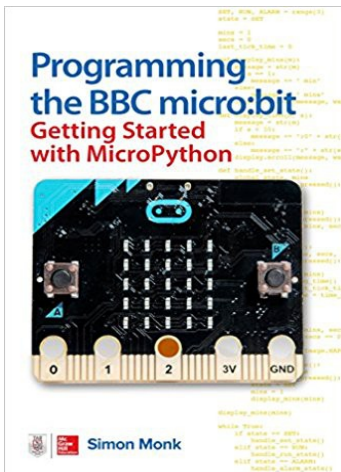
# LEARNING

## micro:bit Programming

If you want to learn more about programming the micro:bit in Python, then you should consider buying Simon Monk's book 'Programming micro:bit: Getting Started with MicroPython', which is available from all major book sellers.

For some interesting project ideas, you might also like micro:bit for the Mad Scientist.

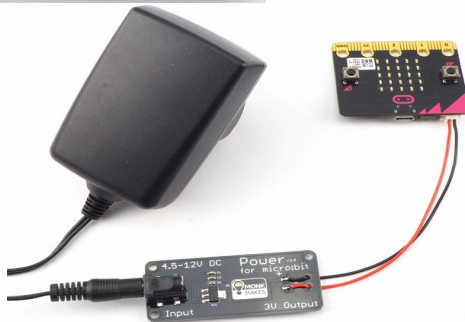
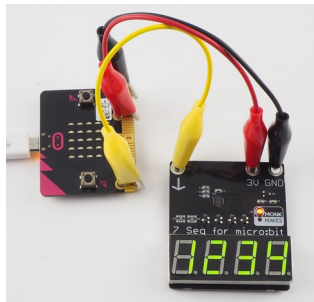
You can find out more about books by Simon Monk (the designer of this kit) at: <http://simonmonk.org> or follow him on Twitter where he is @simonmonk2



# MONKMAKES

For more information on this kit, the product's home page is here:  
[https://monkmakes.com/mb\\_servo\\_kit](https://monkmakes.com/mb_servo_kit)

As well as this kit, MonkMakes makes all sorts of kits and gadgets to help with your maker projects. Find out more, as well as where to buy here:  
<https://monkmakes.com> you can also follow MonkMakes on Twitter @monkmakes.



From left to right: Electronics Starter Kit for micro:bit, Power for micro:bit (AC adapter not included) and 7 Segment for micro:bit.