## Lesson 31 – Activity Sheet

## Setting the Scene

Before you start these activities ensure that your micro:bit is connected to the Kitronik Servo:Lite and that the **servo** motor is connected correctly to the left side pins. All of the programs and code are written to control one servo motor on Pin 1. Pixel 1 on the board may be turned on when running the programs, this is to indicate which pin the servo is connected to. **If you add an additional servo on the right-hand pins, then adapt the code by replacing ‘pin1’ with ‘pin2’.**

Also ensure that the servo has one of the attachments connected. This will make it easier to determine the direction that the motor is turning. At this stage it doesn’t matter which one as it is used simple to track the direction of movement.



[This Photo](http://www.flickr.com/photos/41898857@N04/9367415528/) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)

Move the attachment round until it is pointing in the upwards direction as shown in the image on the left. This will ensure that you can measure a correct half, quarter or full turn. Move the attachment to this position before testing each program.

## Getting Started

## **The Servo Movements**

When writing the code to control the movement of the servo using pin1.write\_analog(200), you need to add a number to indicate the direction. The number for each direction is shown below.

* 100 = 1 millisecond pulse – moves right
* 200 = 2 millisecond pulse – moves left
* 150 = 1.5 millisecond pulse – moves towards the centre

The speed of the servo motor is controlled using the code pin1.set\_analog\_period(10) where the 10 represents 10 milliseconds. A tenth of a second, so the motor turns fairly fast. If you set it to 100 then it hardly moves.

## **Turning the Servo to the Right**

This simple start program turns the servo to the right. Copy the program code below and download to your micro:bit. Turn the Servo:Lite board on and it will power the micro:bit and the motor.

pin1.set\_analog\_period(10)

while True:

pin1.write\_analog(100)

* Notice that the motor appears to speed up, why?
* Can you adapt the program so that the motor turns to the left? pin1.write\_analog(200)
* What happens if you change the pin1.write\_analog(100)to 125, 110 or 135?
* What happens if you set the pin1.set\_analog\_period()to 50?

## **Stopping the Servo Turning**

At some point you will want the motor to stop or only turn for say, five turns. One method is to use Button A as a stop button. If it is pressed the motor stops. This uses the code line pin1.write\_analog(0)where the zero represents no turning and so the motor stops. The other method is to use iteration where the number of turns is set at the beginning of the program and then the program iterates that number of times and then stops the motor.

**Using a Button Press to Stop the Servo**

The program uses a while True loop which keeps the program running, the speed of the turns is also set to 50, which is fairly slow. You can adjust this if you want to. The program keeps checking for Button A being pressed. If it is not pressed then it uses the code pin1.write\_analog(100)to turn the servo to the right. When Button A is pressed then the code pin1.write\_analog(0)stops the servo turning. Copy out the program try it out on the micro:bit.

from microbit import \*

pin1.set\_analog\_period(50)

while True:

if button\_a.is\_pressed():

pin1.write\_analog(0)

break

else:

pin1.write\_analog(100)

* Change the button from A to B, stop that the servo stops when Button B is pressed or even when the micro:bit is shaken

**Using Iteration to Stop the Servo**

This program uses the time module to add a small pause and the module is imported on line two. Next you set the number of turns that the servo will make, in this program it is set to 5. Next the turning speed is set to 50 before a while True loop keeps the program looping.

Next the program checks if the ‘turns’ is greater than 0, it is since you set it to 5, so the program sends the code to rotate the servo to the right. The code sleep(2000) is used to add a two second pause before one is subtracted from the *turns* variable making the new value of turns, 4.

When the program loops round *turns* is again compared to see if it is greater than 0 and it is, since it is now 4.

However, after a few more loops of the program, *turns* will be zero and then the code pin1.write\_analog(0) runs and the servo stops turning. Copy out the program code below and download to your micro:bit.

from microbit import \*

import time

turns = 5

pin1.set\_analog\_period(50)

while True:

if turns > 0:

pin1.write\_analog(100)

sleep(2000)

turns = turns - 1

else:

pin1.write\_analog(0)

If you set *turns* to 10 then the servo will rotate 10 times. **However,** there is a relationship between the turning speed and the pauses. If the sleep() value is not high enough for the servo turn speed to make a complete rotation then the program will start the iteration process before the full rotation has completed and the servo will stop before it makes the full 5 turns.

* Work out the correct time delay for one rotation. At the current settings it is probably about three seconds, so set the delay to sleep(3000).

## Success Criteria

* Program the servo to turn Left
* Program the servo to turn Right
* Adjust the speed that the servo turns at
* Try one of the servo stopping programs

## Pro-tip

* Remember to always move the attachment back the start position that you have selected. This will ensure that the movement of the servo is accurate each time
* The Servo:Lite board can be powered by batteries which means that the motor can be used away from the computer and embedded in other objects such as a Lego car or a robot

## Test Time

Simply download the programs to the micro:bit and try them out. Adjust the values for the speed that the servo attachment turns and also the angle.

## Stretch Tasks

* Write a program so that when a user presses Button A the servo rotates to the left and when they press Button B it rotates to the right
* One possible solution is on the next page

# Add your Python code here. E.g.

from microbit import \*

import time

pin1.set\_analog\_period(10)

while True:

if button\_a.is\_pressed():

display.clear()

pin1.write\_analog(200)

sleep(1000)

elif button\_b.is\_pressed():

display.clear()

pin1.write\_analog(100)

sleep(1000)

else:

display.show(Image.ASLEEP)

* Edit the program code so that the servo rotates only once in either direction

## Final Thoughts

Servo motors are very useful as the speed can be adjusted as well as the direction of rotation. This makes them perfect for use in robots and animatronics. In the next lesson you will look at how to use the servo and the attachments to make a waving hand, a wagging tail, a head that pops up, basically any movement idea that you can think of.