## Lesson 22 – Activity Sheet

Setting the Scene

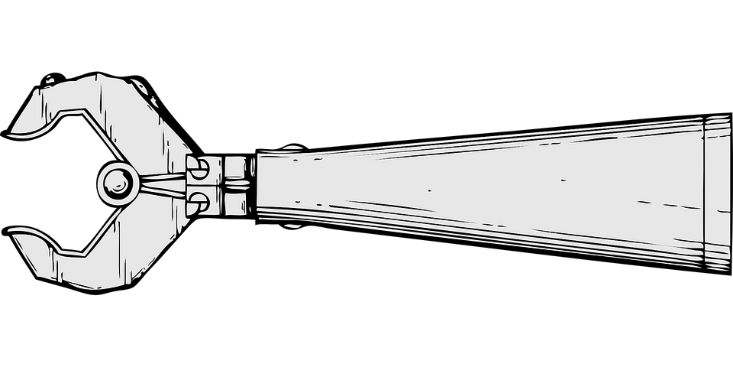
Before you start these activities ensure that your micro:bit is connected to the Kitronik **Servo**:Lite and that the servo motor is connect correctly to the left side pins. All of the programs and code are written to control one servo motor on Pin 1. Pixel 1 on the board may be tuned on when running 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’**

These activities will show you how you can use a servo to push, pull and move other objects. Once you understand how this works then you can adapt the program code to develop you own **animatronics**.

First, ensure that the servo arm attachment is screwed into place, as shown in the image.

Then cut out the robotic arm on the left and fix it to the servo arm. The first program is going to move the robotic arm.

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.





[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/)

## Getting Started

## **Moving the Robotic Arm Left and Right**

The simple program below sends a pulse to the servo which moves the arm to the centre, then after a short delay it moves to the right and then to the left. The code is held in a while True loop which keeps the program repeating. The robotic arm continually move left to right. Try out the program code below.

# Add your Python code here. E.g.

from microbit import \*

import time

while True:

pin1.set\_analog\_period(10)

pin1.write\_analog(150)

sleep(1000)

pin1.write\_analog(100)

sleep(1000)

pin1.write\_analog(150)

sleep(1000)

pin1.write\_analog(200)

sleep(1000)

## **The Servo Movements**

When writing the code to control the movement, for example pin1.write\_analog(200), of the servo 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 fair fast. If you set it to 100 then it hardly moves.

* Edit the program so that the robotic arm moves to the left fast and then to the right, but slowly

**Using the Buttons to Move the Robotic Arm Left or Right**

This next program enables you to move the arm left or right using Buttons A and B. It also displays and image whilst the arm is not moving. Copy and download the program code below.

# Add your Python code here. E.g.

from microbit import \*

import time

pin1.set\_analog\_period(10)

while True:

if button\_a.is\_pressed():

Since the micro:bit displays an image when the Arm **is not** moving, it needs to be cleared when it is. This uses the code display.clear()

To stop the servo moving the program uses the code pin1.write\_analog(0)

display.clear()

pin1.write\_analog(200)

sleep(2000)

pin1.write\_analog(0)

elif button\_b.is\_pressed():

display.clear()

pin1.write\_analog(100)

sleep(2000)

pin1.write\_analog(0)

else:

display.show(Image.ASLEEP)

* Change the image that is displayed
* Edit the program so that the Robotic Arm moves to at a different speed
* Display a different image, maybe the left arrow, when the arm moves to the left

**Program a Quarter Turn**

Most projects that use servos require a precise movement, which is why they are used. This next program adjusts the values in the previous program to create a quarter turn. If Button A is pressed, then the robotic arm moves only a quarter to the left. If Button B is pressed, then the arm moves only a quarter to the right.

This precision enables you to accurately program leg movements; arm movements; a mouth opening and closing, yawning or talking. The possibilities are endless. By reducing the sleep time to 250 it stops the arm turning before it completes a full turn. Make the edits as shown below and try out the program.

# 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(250)**

pin1.write\_analog(0)

elif button\_b.is\_pressed():

display.clear()

pin1.write\_analog(100)

**sleep(250)**

pin1.write\_analog(0)

else:

display.show(Image.ASLEEP)

* Change the image that is displayed
* Change the amount of turn to a half turn. (Remember to move the servo back to the centre first)
* Program the arm to return to the centre once it has completed the turn

## Success Criteria

* Attach the arm and robot arm image to the servo
* Program a simple code to move the robot arm left and right
* Program the servo to turn left and right when a button is pressed
* Program a quarter turn of the robot arm
* Program a half turn of the robot arm
* Use the program code and attachments to build your own animatronic

## 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 for the angle.

## Stretch Tasks

Now that you can control the direction, speed and amount of turn of the servo you can use it to build your own animatronic. Look at the available attachments and decide which one will best support the movement. The small holes in the attachments can be used to connect wires or paperclips which can connect to something else, for example a cardboard flap.

|  |  |
| --- | --- |
| **Movement Ideas** | **Servo Use** |
| A waving hand | Stick an image of a hand to the arm attachment |
| A mouth opening and closing | Attach a wire from the mouth to the servo attachment, as it turns it pulls the mouth down. Turn the Servo in the opposite direction to close the mouth. |
| A Lego figure moving towards and away from you | Attach a lollipop stick to the arm. Place a Lego figure at the end. As the arm rotates it moves the lollypop stick back, moving the figure away. |
| A globe | Paint a ping-pong ball like the Earth or another planet. Use the circular disc attachment and place the earth / planet on top. Program the motor to turn the planet. Research the speed and add to the program code to make it more accurate. |

## 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. The skill and techniques you have learnt in this lesson will be useful in the micro:PET project.