IoT4x Module 1 - Tinkercad® Activity 2

Tinkercad® Circuits features a simulated Arduino Uno microcontroller board. This can be used to read from sensors and control actuators once the microcontroller has been programmed to do so. The simulated Arduino can be programmed in a visual Scratch-based language or a textual C-based language called Wiring.

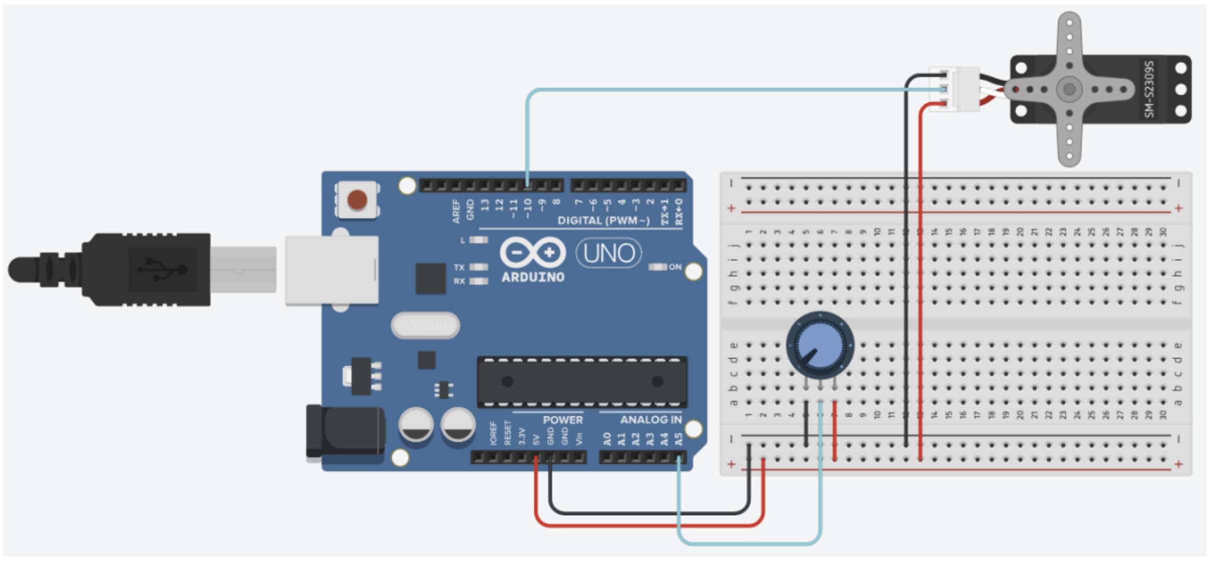
If you are not already familiar with how to use Tinkercad® circuits then watch the Introduction to Tinkercad® video before following these steps:

1. Log in to your account on the Tinkercad® website. Tinkercad® runs best using Google Chrome as the browser.

2. Click the following link to open a circuit in Tinkercad® which has already been created for you https://www.tinkercad.com/things/17ya9oiulVc

3. Click on **Copy and Tinker** to create a copy of the circuit in your own account.

The circuit will contain an Arduino Uno microcontroller board, a breadboard, a potentiometer and a micro servo. A screenshot image is below.



You may already be familiar with these components from the IoT2x, Sensors and Devices course or elsewhere. If not, there is no requirement in this course to understand in detail how they work, but here is a brief description of the function of these components:

- The potentiometer is the blue dial on the breadboard. It is a resistor with a resistance that varies in value as the dial is adjusted. It is currently wired to an analogue input (A5) of the Arduino. The changing resistance will result in a voltage between 0 and 5V being applied to the A5 input depending on the adjustment of the dial. The Arduino's analogue to digital converter will convert this analogue input voltage into a discrete digital reading ranging from 0 to 1023.

- The “SM-S2309S” micro servo is the device sitting above the breadboard. A servo is an electric motor that can be rotated to stop in specific positions. In this caseifferent signals from the microcontroller tell the servo to rotate the shaft to any of the positions from 0 to 180 degrees.

Program the Arduino to Move the Servo

Note:

- *When using code blocks in Tinkercad®, if any blocks are accidentally deleted you can restore them by right-clicking on the rubbish bin symbol and choosing “Undo”.*
- *Occasionally a glitch on the website means that blocks are deleted when they are moved. If this happens then restore your blocks using “Undo” and refresh your browser page.*

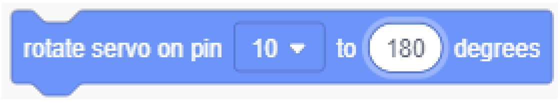
1. Click the **Code** button near the upper-right of the Tinkercad® screen.
   The code window will slide out from the right-hand side of screen. Although the components have already been wired you will see that the microcontroller has not yet been programmed.

2. Click the drop-down list to change code view from **Blocks** to **Blocks + Text**. You will see that the text equivalent code is currently an empty template of a “setup” function and a “loop function with a default 10 millisecond delay.

3. From the blue **Output** code block menu, find the **rotate servo on pin 0 to 0 degrees** block and move it into the block code space. You will see some changes in the text code.

4. The **Signal** wire of the servo connects to digital pin 10 of the Arduino. Click the pin number on the servo code block and select **10** from the drop-down list of pins.

5. The servo in this simulation always begins at its 0 degrees position. Change the number in the code block to **180** so that it will move the servo across its full range of rotation to the 180 degrees position.



“Autodesk screen shots reprinted courtesy of Autodesk, Inc.”

6. Click the **Start Simulation** button and you should see the servo rotate to a new position. You may need to resize or close the code window to see the servo and restart the simulation.

Introduce Variables to the Code

We are now going to program the servo to continuously rotate from the 0 degrees position to the 180 degrees and back again, while briefly stopping at every single degree position in-between.

1. In the code window, click the pink **Variables** menu option.

2. Click the **Create variable…** button. A window will pop up asking for the new variable's name.

3. Call this variable **angle** by typing in the new name and pressing **OK**. You will see the new pink code block appear in the menu which represents the new variable. Two other blocks will also appear which allow you to set and change the value stored in the variable.

4. Drag the **angle** variable on to the existing servo code block above the number 180. It should snap in to place so that the block now reads **rotate servo on pin 10 to angle degrees**. If it doesn't snap in then try clicking and dragging it to a slightly different position before dropping it.

When the code caused the servo to rotate to 180 degrees, the number 180 was a "constant". This meant it did not change at any point during the execution of the code. By replacing the setting for degrees with a variable that represents a number, it can be modified so that it potentially represents a different number each time this code block is encountered. We now need to ensure other parts of the code can modify the value of "angle".

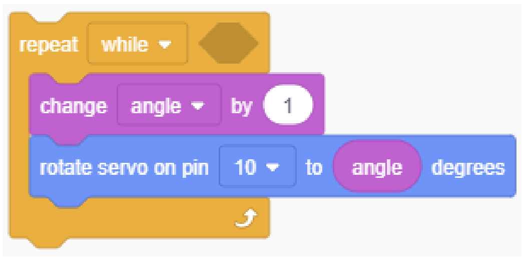
5. Staying in the pink **Variables** menu, find the **change angle by 0** code block and drag it so that it connects above the existing blue servo code block. Notice the changes in the text code.

6. Click and delete the **0** in this new code block and change it to **1** so that it now reads **change angle by 1**.

You can see in the first lines of the text code that the variable called "angle" begins with an initial value of "0". Running the code now would increment this this from "0" to "1" before rotating the servo to the "1 degree" position. Continuing over these lines of code a second time would increment it from "1" to "2" before rotating the servo to the "2 degree" position, and so on.
We are now going to set up a loop that rotates the servo sequentially to each degree between 0 and 180.

7. From the orange **Control** code block menu, move the **repeat while** block into to the block code workspace.

8. Move the **repeat while** block over the other two code blocks so that it wraps around them as shown below. You may need to rearrange blocks if it doesn't look exactly like this the first time.


```
repeat while ◆
  change angle by 1
  rotate servo on pin 10 to angle degrees
```

"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

9. From the green **Math** menu, find the **1 < 1** block and drag it so that snaps into the hexagon shape in the orange block so that it becomes **repeat while 1 < 1**. This does not make sense and will never be "true" but we will now modify this line.

10. From the pink **Variables** menu, take another copy of the **angle** variable block and place it over the left-hand **1** in the green block.

11. Change the right-hand **1** in the green block to **180** so that the line has become **repeat while angle < 180**.

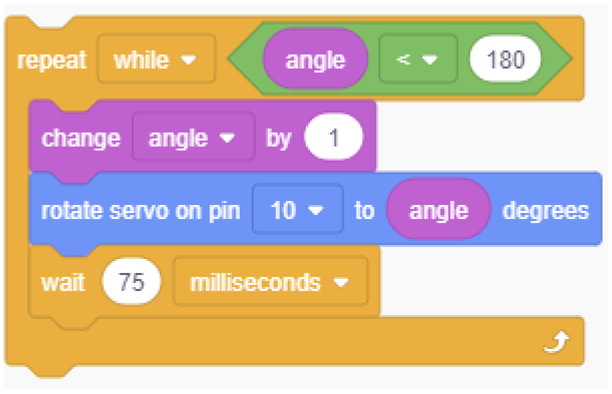
We know that every time the code inside this block runs it will increment the angle variable by 1 before moving the servo to the corresponding position. This execution of the code blocks inside the orange loop will repeat until "angle" reaches the value of "180". After this, there are no other lines of code which will modify the value of "angle" so the code within the loop will not execute again (until the simulation is restarted).

12. From the orange **Control** code block menu, select the **wait 1 secs** block and drag it below the blue servo code block so that it becomes the last line of code inside the loop.

13. Edit this code block from **wait 1 secs** to become **wait 75 milliseconds**.

The code in the loop can execute faster than the motor in the servo can physically move so by adding this small delay we give the servo time to keep up with the code.

The code should now look like the screenshot below.



```
repeat while angle < 180
    change angle by 1
    rotate servo on pin 10 to angle degrees
    wait 75 milliseconds
```

"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

Running this code will cause the servo to slowly sweep from 0 to 180 degrees. We will now add a second loop which will also allow it to sweep back to the 0 degree position.

14. Hold your mouse cursor over the word **repeat** at the beginning of the orange code block and right-click it so that a shortcut menu appears.

15. Click the **Duplicate** option from the menu and a copy of the entire code loop and all its contents will appear attached to your mouse cursor.

16. Connect the new block to the bottom of the original one so that it becomes an identical new loop below the original loop.

17. In the green code block of the new, second loop change **angle < 180** to instead be **angle > 0**. i.e., change the **less than** sign to a **greater than** sign and change the number from **180** to **0**.

18. Within the second loop, change the **change angle by 1** block to **change angle by -1**.
    i.e., change the **1** to **-1** so that it decrements rather than increments.

19. Click the **Start Simulation** button and watch the servo arms slowly sweep between the 0 and 180 degree positions.

Looking at the text-based equivalent code, you can see that the movement of the servo is actually being controlled by three loops altogether, with two loops inside a third.

The clockwise rotation occurs when the code is in the loop under the condition, **while (angle < 180)** *and the anti-clockwise rotation occurs when the code is in the loop under the condition,* **while (angle > 0)**.

Both of these loops sit within the third loop, **void loop ()** *so that the servo continues in this pattern until the simulation is stopped or, in reality, power is removed from the Arduino.*

Read From an Input to Change the Output

At this point the potentiometer is correctly wired to an input of the Arduino but if you were to change the dial while the simulation is running you would find it has no effect. There is nothing in the code telling the microcontroller to read from the potentiometer at the input

We are now going to add code so that the potentiometer will control the movement speed of the servo.

1. Go back to the pink **Variables** menu and create two more variables. One should be called **reading** and the other **multiplier**.

2. From the pink **Variables** menu, drag the **set angle to 0** block into the code block area but leave it separate so that it does not connect to other blocks.

3. Repeat step 2 by dragging a second **set angle to 0** block but connect it to the one you just created. These two pink blocks should remain separate from the loop blocks at this stage.

4. Click the drop-down menus on the two new pink blocks and change the variable from the first one to **reading** and the second one to **multiplier**.

5. Click on the purple **Input** code block menu and find the **read analog pin 0** block. Drag it into the first pink block to replace the **0**. This line of code should become **set reading to read analog pin A0**.

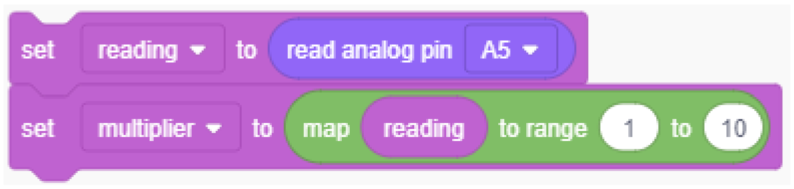
6. Change the **A0** to **A5** so that this block will now read from the analogue input pin that the potentiometer connects to and save this to the **reading** variable.

7. From the green **Math** menu, find the **map 0 to range 0 to 180** block and drag it into the second pink block to replace the **0**. It should now read **set multiplier to map 0 to range 0 to 180**.

8. From the pink **Variables** menu, drag a new **reading** variable block and use it to replace the first **0** in the green block that was placed in the previous step.

9. The default range in the green block is **"0** to **180**". Modify it to become "**1** to **10**".

The two new lines of block code should now look like the screenshot below.



```
set reading to read analog pin A5
set multiplier to map reading to range 1 to 10
```

"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

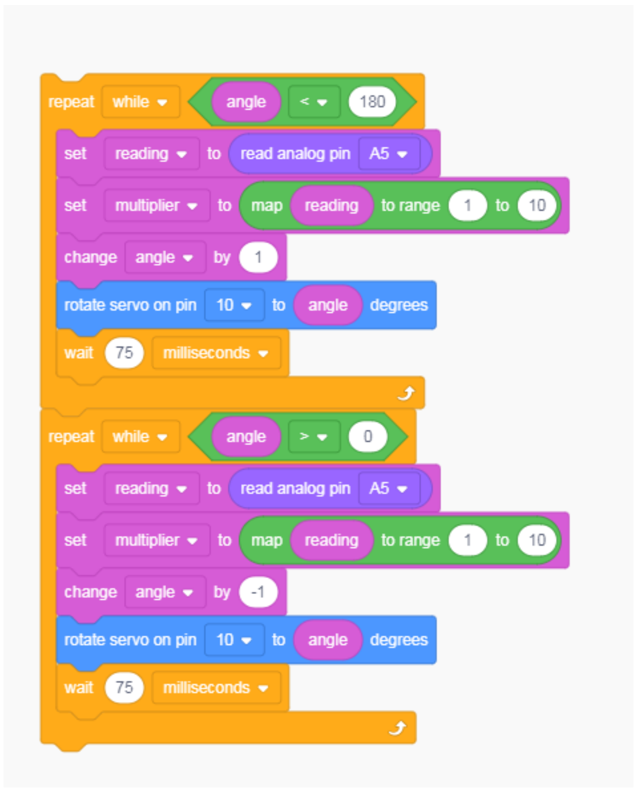
These lines of code will read the analogue voltage from the potentiometer, convert it to a digital reading which will be anywhere between 0 and 1023 inclusive (as mentioned at the beginning of this activity) and store it in the variable called "reading".

Next, it will convert the reading to its equivalent number in a different range between 1 and 10 inclusive before storing it in the "multiplier" variable.

10. Right-click on the word **set** on the first of these two new pink blocks and click **Duplicate**. An identical copy of the two blocks will now be attached to the mouse cursor.

11. Place this duplicate pair of pink blocks within the first loop under the **repeat while angle < 180**. They will become the first two lines of code of the first loop, i.e., there should now be five lines of block code within that loop (see the screenshot below if unsure).

12. Place the other pair of pink blocks within the second loop under the **repeat while angle > 0**. They will become the first two lines of code of the second loop, i.e., there should now be five lines of block code within that loop (see the screenshot below if unsure).



```
repeat while angle < 180
    set reading to read analog pin A5
    set multiplier to map reading to range 1 to 10
    change angle by 1
    rotate servo on pin 10 to angle degrees
    wait 75 milliseconds
repeat while angle > 0
    set reading to read analog pin A5
    set multiplier to map reading to range 1 to 10
    change angle by -1
    rotate servo on pin 10 to angle degrees
    wait 75 milliseconds
```

"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

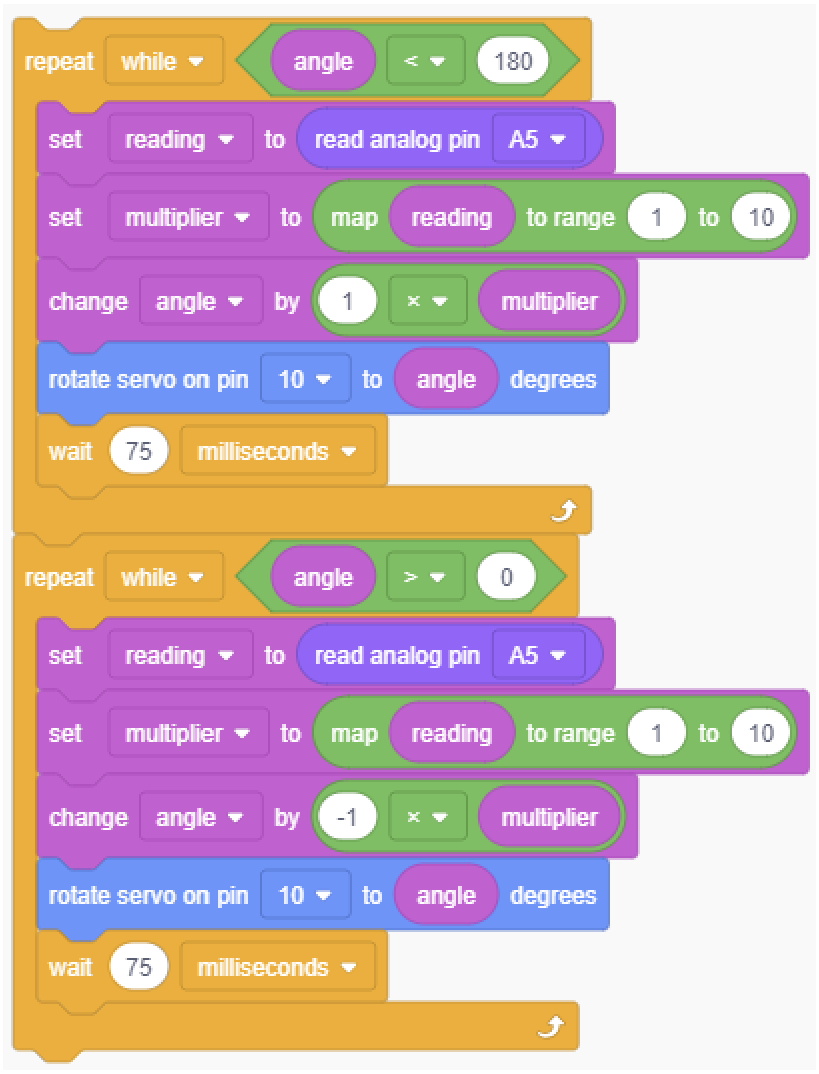
Each loop now reads from the potentiometer as an input then converts it to a range for use at the output. Now we just need to modify the code so that the input will affect the output.

13. From the green **Math** menu, find the **1 + 1** block and drag it to the loop so that it replaces the **1** in the **change angle by 1** block.

14. Drag a second **1 + 1** block to the other loop so that it replaces the **-1** in the **change angle by -1** block.

15. Using copies of the **multiplier** variable from the pink **Variables** menu, make changes to the green blocks added in the previous two steps as follows:
    - First loop: modify **1 + 1** so that the line becomes **change angle by 1 x multiplier**
    - Second loop: modify **1 + 1** so that the line becomes **change angle by -1 x multiplier**

The code should now look like the screenshot below



```
repeat while angle < 180
    set reading to read analog pin A5
    set multiplier to map reading to range 1 to 10
    change angle by 1 × multiplier
    rotate servo on pin 10 to angle degrees
    wait 75 milliseconds
repeat while angle > 0
    set reading to read analog pin A5
    set multiplier to map reading to range 1 to 10
    change angle by -1 × multiplier
    rotate servo on pin 10 to angle degrees
    wait 75 milliseconds
```

16. Click the **Start Simulation** button and click and drag the dial on the blue potentiometer to different positions. You should see that it increases and decreases the speed of rotation.

- *Each time the code is executed a reading between 0 and 1023 is taken from the potentiometer at the A5 input, stored in the variable called "reading", converted to its equivalent number in a range between 1 and 10, which is then stored in the "multiplier" variable.*

- *The "multiplier" variable affects the speed of rotation. If you don't understand how, then try calculating how much the "angle" variable will change each time the **change angle by…** blocks are run but substituting different values of "multiplier" between 1 and 10.*

- *To help understand the operation of any program with variables, try manually stepping through each iteration of the code on paper, noting down what the values of variables would be each time they change as each line of code is executed.*

END OF ACTIVITY 2