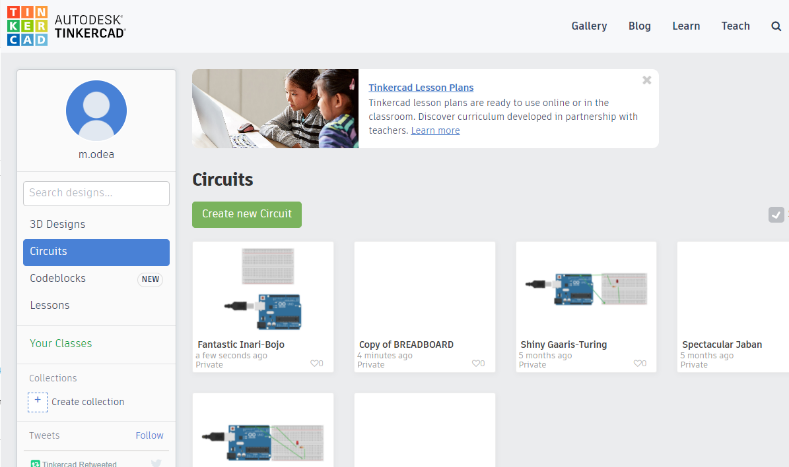
**Welcome Week - Tinkercad® Activity**

*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.*

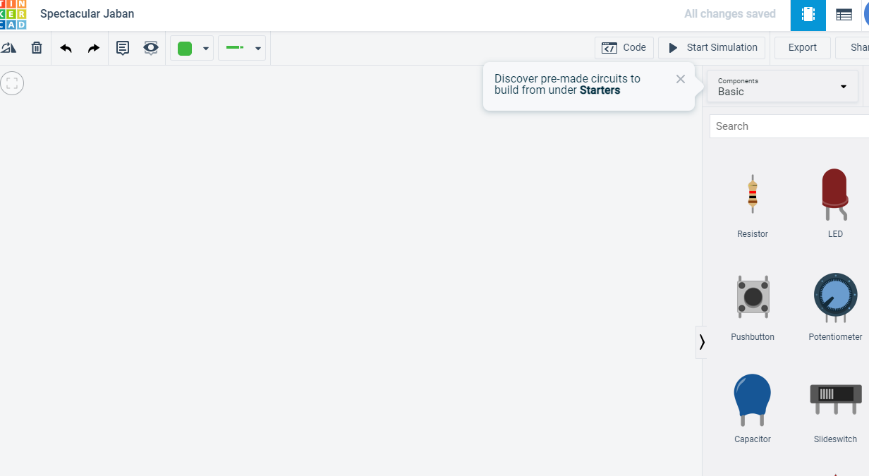
Go to the following pages and create an account – use your university email address. <https://accounts.autodesk.com/register?viewmode=iframe&ReturnUrl=%2Fauthorize%3Fviewmode%3Diframe%26lang%3Den-US%26realm%3D%252A.tinkercad.com%26ctx%3Dtinkercad%26authtype%3Dsocialoradsk%26socialproviderpref%3DFB%2CGO%2CYH%2CWL%2CAP%26AuthKey%3Dc611256a-4727-42dd-8a1d-63ba0083c295>

Then go to here <https://www.tinkercad.com/dashboard> and click on create new design.

Select Circuits from the lefthand side and then create new circuit



Then you should get a screen like this



Select Components all here

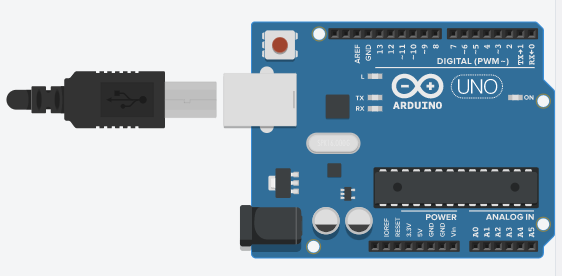
Now follow the instructions below.

Note: If you are not already familiar with how to use Tinkercad® circuits then watch the <https://www.youtube.com/watch?v=HdHli3ZuIa0> video before following the steps below where you will program a microcontroller to light an LED in response to the press of a button.

# Place the Microcontroller and Breadboard

1. Look through the **Components** menu and find an **Arduino Uno R3**. Place one of these in the workspace. This is a circuit board with a microcontroller and sockets (or pins) for easily connecting to inputs and outputs.

It provides power and the ability to provide a controlled power supply to a breadboard.



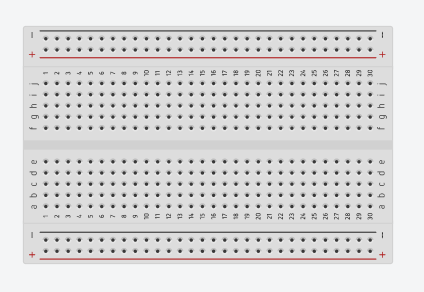
The important elements of the Arduino board are as follows:

Power – shown in blue above. Arduino has 5v, 3.3v and ground.

Programmable digital pins – shown in red above

Programmable analog pins – shown in green above

1. Also take the **Breadboard Small** from the menu and place to the right of the Arduino. A *solderless breadboard* has a number of sockets (or pins) for easily plugging in electronic components and wires. Some of these pins are connected to each other internally.



A breadboard works in the following way.

The outer rails (outlined in blue above) labelled + and - are connected vertically together, shown in green. This means anything connected along the same vertical rails is physically connected.

The inner rails labelled a – j above run horizontally as shown above in red.

To place a component on the circuit, select it from the menu and click and drag onto the breadboard.

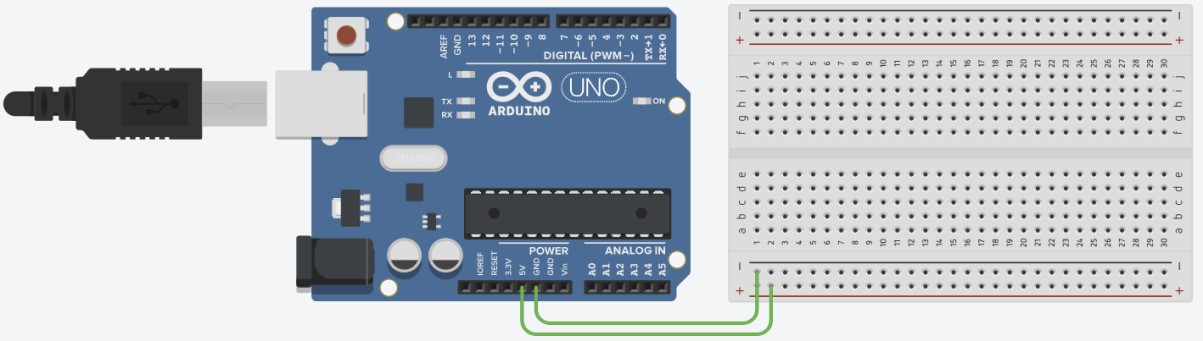
To connect components click on a pin where the wire starts and then click again on the pin to where the wire ends.

Here is an example

1. Click and drag to make a wired connection between the **5V** (5 Volts) socket at the bottom of the Arduino and the positive (red) rail near the bottom of the breadboard. See screenshot below for reference. This will connect all pins along that row to the 5 Volts source from the Arduino Uno board so that any may be used for this purpose.

1. Click and drag to make a wired connection between any **GND** (ground) socket of the Arduino and the negative (black) rail near the bottom of the breadboard. There are three sockets on the Arduino board labelled **GND** and any of these can be used for the same purpose. See screenshot below for reference. This will connect all pins along that row to the zero Volts reference from the Arduino Uno board so that any may be used for this purpose.

*Your circuit should now look similar to the screenshot below*



“Autodesk screen shots

reprinted courtesy of Autodesk, Inc.”

***Note:*** *In all Tinkercad® activities and assessments in this course the supply voltage used will be 5 Volts. When dealing with digital inputs or outputs, any reference to “1”, “logical 1”, “on” or “HIGH” relates to the presence of this 5V supply voltage. Any reference to “0”, “logical 0”, “off”, “LOW”, “GND” or “ground” relates to an absence of the supply voltage.*

Exercise 1: Make a simple LED circuit

For this exercise you will need an LED and a resistor.

1. Connect a 3.3v power supply from the Arduino to an outer rail that is labelled +
2. Connect a ground to another outer rail, one labelled -.
3. Add an LED onto the breadboard inner rails. You may need to rotate this to get the different pins onto 2 separate rows.
4. Add a resistor onto the breadboard – you may also have to rotate this as well.
5. Connect the 3.3v supply to one side of the resistor.
6. Connect the other side of the resistor to the positive (Anode) pin of the LED.
7. Connect the negative (Cathode) side of the LED to the ground.
8. Test the circuit by pressing the START SIMULATION button. The LED should light up.

# Exercise 2: Create a basic programmable circuit

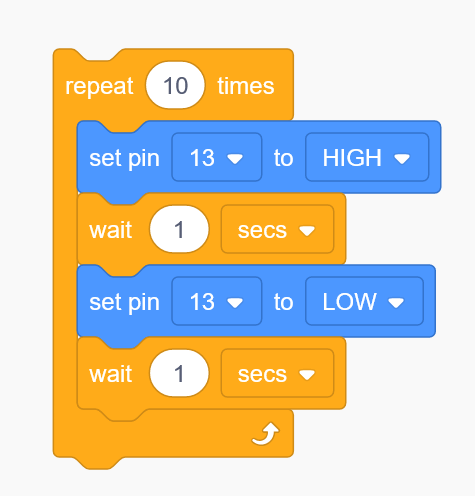
Copy the circuit from Exercise 1 and update the circuit so that the power from the circuit comes from pin 13 and not 3.3v.

This will allow us to do things like turn the power on and off by controlling pin 13.

Now we are going to add some programming from the Arduino to control the circuit.

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 and you will see some of the blocks that can be used for programming the Arduino. The border between this window and the workspace can be adjusted by hovering your mouse over it, clicking and dragging.

You will also see four blocks which have been left in place as an example.

1. From the coloured menu that lists block categories (screenshot pictured here), select **Output**. All the blue coloured output blocks will appear in the menu.
2. Select the block **Set Pin …** and drag and drop. Update the control to pin 13 and output to HIGH, this will allow us to set pin 13 as an output, i.e. enable it to send power to the circuit when we program it to do so.
3. Add a second update it to 13 and set the output to LOW. This will turn power on.
4. Now add a control block. Select a repeat block.
5. Connect them all as shown in the diagram below.  
     
   
6. At the top of the code window is a drop-down list which will currently show **Blocks**, which means it is showing the visual block code only. Click on the list to open it and select the **Blocks + Text** option. You will see an extra screen appear which will show the text equivalent of the block code. This text code is what will actually be compiled and uploaded to the Arduino.

Now run the simulation.

# Exercise 3: Connect a Basic Sensor and Actuator

1. From the **Components** menu, place a **Pushbutton** on the breadboard so that its upper pins sit in row ‘f’ and its lower pins sit in row ‘e’ (as labelled on the edge of the breadboard). It doesn’t matter which column you place this in. Use the screenshot below for reference if needed.

*The Pushbutton can be classed as a sensor in this circuit because it senses whether the button is being pushed or not based on whether there is an electrical connection between two of its pins.*

1. Connect a wire between one of the pushbutton’s lower pins and the positive rail. See the screenshot below for reference if needed.

1. Connect a wire between the other one of the switch’s lower pins and pin 7 of the Arduino.

1. Place an **LED** in the breadboard so its legs sit in row ‘e’.

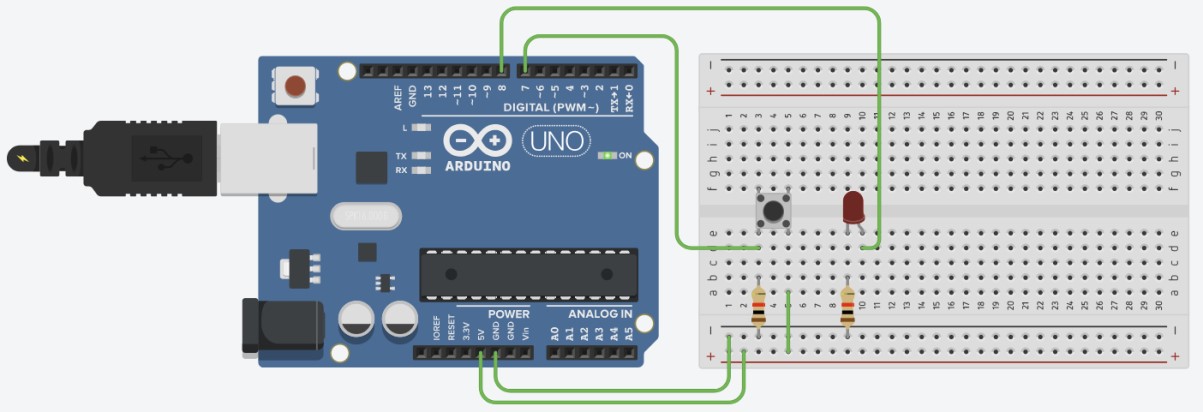
*The LED represents the actuator in this circuit because its state can be modified (brightness, on or off) by a signal from the microcontroller.*

1. Connect a wire between the **anode** of the LED (the bent leg) and pin 8 of the Arduino. You can check that you have the correct leg of the LED by hovering your mouse cursor above it and reading the label.

1. Take a **Resistor** from the **Components** menu, leave it at its default value of 1kΩ and use it to make a connection between the **cathode** of the LED (the other leg) and the negative rail. This limits to prevent excessive current through the LED.

1. Take another **Resistor** from the **Components** menu, also leave it at its default value and use it to make a connection between the pushbutton pin that connects to the Arduino and the negative rail. This will improve the accuracy of the signal at the Arduino input pin by guaranteeing the voltage is at 0 when the button is not being pushed.

*Your circuit should now look similar to the screenshot below*



“Autodesk screen shots

reprinted courtesy of Autodesk, Inc.”

# Program the Microcontroller to Read the Input

1. Click and drag the **read digital pin 0** block into the code block space to the right of the block menu. Notice that there are two different new lines of the text code added for this block.



1. We previously connected the Pushbutton switch to digital pin 7 of the Arduino. Click the section of the block containing the ‘0’ and a list will drop down. Select ‘7’ so that the block now displays **read digital pin 7**.

*The switch is “normally open”, so that no connection is made between its pins and no voltage will normally be present at the input. The Arduino will interpret this digitally as a ‘0’ or ‘LOW’ when reading the pin. When the button is pushed it closes the circuit and the 5V from the Arduino is connected to the input. The Arduino will interpret this digitally as a ‘1’ or ‘HIGH’ when reading the pin.*

# Exercise 4: Program the Microcontroller to Write to an Output

*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 will disappear and be deleted whenever they are clicked. If this happens, you should restore your blocks using “Undo” and then* ***refresh*** *your browser page and it should operate correctly.*

1. From the coloured menu that lists block categories, select **Output**. All the blue coloured output blocks will appear in the menu.

1. Click and drag the **set pin 0 to HIGH** block into the code block space below the input block we placed. Keep it separate from the other block so that it doesn’t connect yet. Notice there are two different new lines of the text code added for this block.

1. We previously connected the LED to digital pin 8 of the

Arduino. Click the section of the block containing the ‘0’ and a list will drop down. Select ‘8’ so that the block now displays **set pin 8 to HIGH**.

1. Right-click the blue code block and a shortcut menu will appear. Click **Duplicate**. An identical block will appear and follow your mouse cursor.

1. Click to place the duplicate block in the space below the original. Keep it separate from the other blocks so that it doesn’t connect yet.

1. Click the section of the new block containing the ‘HIGH’ and a list will drop down. Select ‘LOW’ so that the block now displays **set pin 8 to LOW**.

*When pin 8 has been set as an output and is set to LOW it will output a digital ‘0’ (approximately 0V). When set to HIGH it will output a digital ‘1’ (approximately 5V).*

*The code for reading inputs and writing to outputs is now in place. The next step is to tell the microcontroller how and when to respond to the input by writing to the output.*

# Exercise 5: Program Logic and Conditions in the Microcontroller

If we want the LED to turn on when the button is pressed, then we need to specify that the button has been pressed whenever a ‘HIGH’ signal is present at pin 7.

1. From the coloured menu that lists block categories, select **Math**. All the green coloured blocks will appear in the menu.

1. Click and drag the green **1 < 1** block into the code block space below the other blocks we already placed. Notice there is one new line of the text code added for this block.

1. Click and drag the purple **read digital pin 7** block to where its left edge is above the first **1** in the new green block and release it so the purple block snaps into the place where the **1** previously was. You will see the green block automatically expand so that it fits.

1. Click and drag a green **HIGH** block from the Math menu and drop it on top of the other **1** in our first green block.

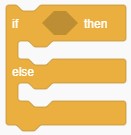
1. Click the **< (less than)** sign in the first green block so that a list of options drops down and change it to an **= (equals)** sign.

The green and purple block should now state **read digital pin 7 = HIGH** as in the screenshot below. Notice the equivalent line in the text code.



1. From the coloured menu that lists block categories, select **Control**. All the orange coloured blocks will appear in the menu.

1. Find the orange block containing all three of the words “**if**”, “**then**” and “**else**”. Click and drag this into some empty block code space.

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

1. To move the combined green and purple block you will need to click and drag it by any of its green parts where your cursor takes the shape of a hand. If you click and drag it at the wrong point it will break apart. Move this block into the new **if, then, else** block and drop it within the six-sided (hexagonal) shape so that it snaps into place.

1. Move the blue **set pin 8 to HIGH** block so that it snaps into place on the orange block between the **if** and **else**.

1. Move the blue **set pin 8 to LOW** block so that it snaps into place on the orange block below the **else**.

*Your block code should now look exactly like the screenshot below*



# Test the Circuit

Notice the equivalent text code that has appeared. It should be identical to the screenshot below.



You don’t have to remember how this text code is written at the moment but you can follow along to gain a general understanding of the functions:

* You can see that the program firstly sets pins 7 and 8 of the Arduino as an input and output respectively.
* It then enters a continuous loop that checks to see whether pin 7 is equal to a HIGH signal.
* If this is true then it will set pin 8 to HIGH.
* If pin 7 was not equal to a HIGH signal it will set pin 8 to LOW.
* The final line of the text code shows a 10-microsecond delay before the loop begins again. This is automatically added by Tinkercad® to improve the simulation’s performance.

1. Click the **Code** button on screen to close the Code window so that you can see the Arduino, breadboard and components.

1. Click the **Start Simulation** button near the top-right of the screen. You will see the plug connect to the simulated Arduino, it will power up and the code you have created will be uploaded to program its microcontroller. Other than this you should not yet see any changes.

1. Test the circuit by clicking the Pushbutton. As you click and hold the button the LED should light up. Releasing the button should let the LED turn off again.

If it does not work as it should then go back and check your circuit and the code against the instructions and screenshots.

***Notes:***

* *This simple “switch and LED” circuit could actually be created to perform the same function without needing the Arduino by wiring it differently and providing a separate power source. However, trying to create more complex circuits and functions could be much more difficult or nearly impossible to create by just using standard electronic components. A microcontroller is a small computer that can be programmed to perform different purposes and be easily controlled or monitored via other devices.*

* *Continuous looping to check whether or not the switch has been pressed (polling) is a simple but inefficient use of this microcontroller. More complex microcontroller programming methods will allow it to perform other tasks the rest of the time and only respond to the switch if it has actually been pressed.*

* *More efficient methods would also be used to minimise power consumption. This would be particularly important when running from battery power. Many microcontrollers can be put into a low powered “sleep” mode and only “awaken” to take action when the button is pressed.*

*“Autodesk, AutoCAD, Tinkercad, DWG, the DWG logo, and Inventor are registered trademarks or trademarks of*

*Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries.”*