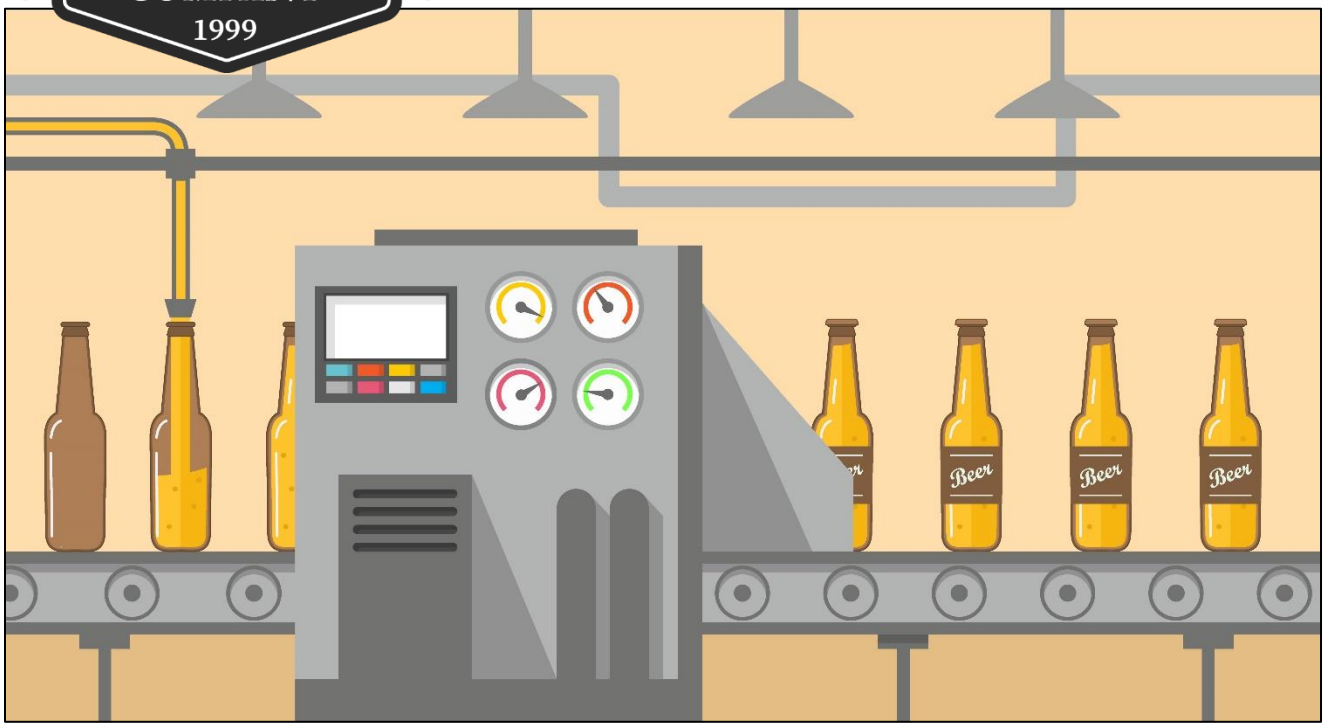
Welcome to the IoT Brewing Company!



Prepared by:
Chuck Bales
Dr. Kristine Christensen

Developed in partnership with CSSIA's IoT Coordination Network.

Introduction – Ready to Clock In?

Welcome to the IoT Brewing Company! You have recently been hired as an apprentice brewmaster at the IoT Brewing Company which makes and bottles craft beers at a downtown facility. The company has decided to adopt smart manufacturing principles and have embraced Internet of Things (IoT) technologies to monitor the brewing process, increase brewing efficiencies and quality standards. You will work with the brewmaster to increase the amount of automation by introducing sensors and actuators that will collect information throughout the brewing and bottling process.

The Internet of Things is a vast array of interconnected devices which transfer data and communicate over a network without requiring human interaction. Common IoT devices include smartphones, fitness trackers, connected thermostats, internet-enabled home security systems, intelligent personal assistants such as Alexa, and many more. IoT devices are composed of a microcontroller such as an Arduino, sensors, actuators, and some type of connection to a network such as the internet. The network connection may be via Bluetooth, Wi-Fi, cellular, satellite, or directly via ethernet. Typically, an IoT device will also include some type of interface such as an app or webpage.

The IoT device you will be creating in this project will be composed of a microcontroller (Arduino Uno), sensors (input devices), and actuators (output devices). Tinkercad does not currently allow connecting to an external network so you will be simulating the IoT device without the connection to a network. However, at the conclusion of the project we will be explaining the steps required to make this a true IoT device.

Brewmaster Apprentice Orientation Activities

Before you can start working as the brewmaster's apprentice you will need to complete a series of activities geared toward orienting you to the tools, components, and programming principles needed to be a successful apprentice.

You will need to complete activities that will help you complete two Orientation projects which include:

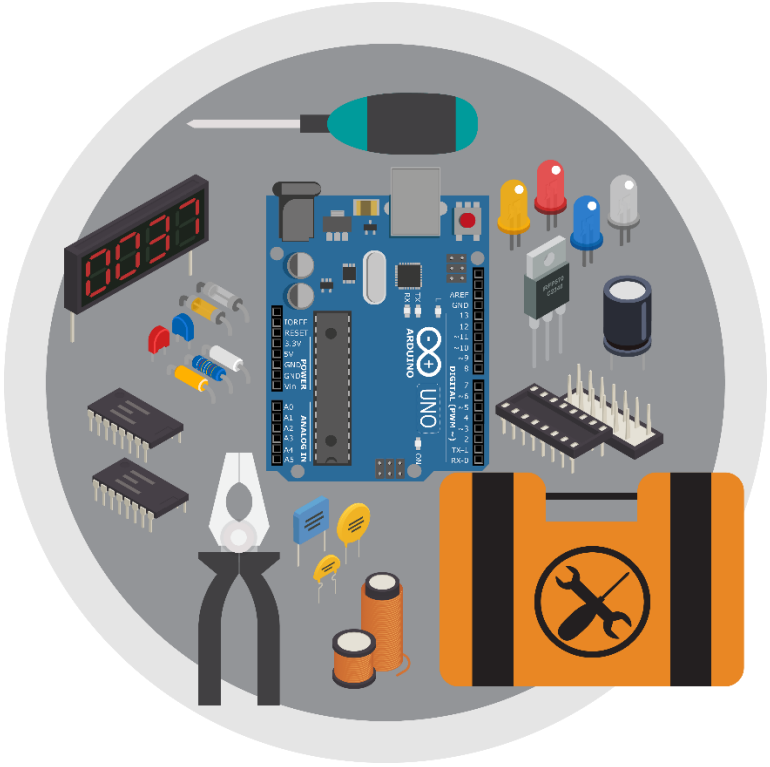
Introductory Activity 1. Registering for a Tinkercad account.

Introductory Activity 2. Review the overview of required components to complete projects.

Project 1. Use a pushbutton switch to control a LED.

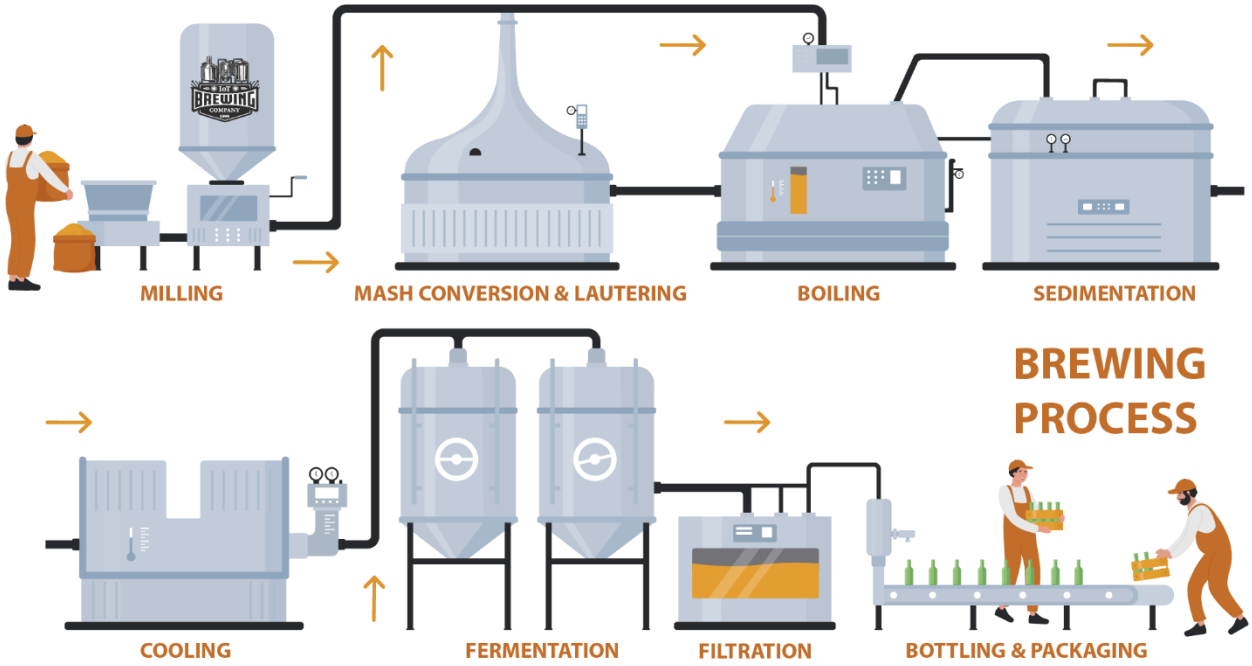
Introductory Activity 3. Review the overview of Arduino Programming.

Project 2. Use a pushbutton switch and programming code to blink an LED.



The Brewing Process

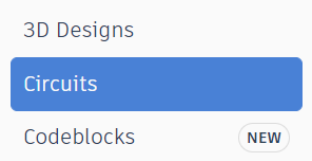
Before you begin your IoT projects you need to understand how the beer brewing and bottling process works. The grains are first milled to break up the kernels to extract the sugars for fermenting. The milled grains are then transferred to a vessel which is mixed with water and heated in a process called mash conversion. The mash is then pumped into a second vessel for lautering where the liquid, called wort, is separated from the grain husks. The wort is then transferred to another vessel called a kettle and is brought to boil prior to adding the hops for flavoring. Once boiled, the wort is transferred to a sedimentation vessel where the wort is separated, and any remaining hops particles are filtered from the liquid. The filtered wort is then transferred to a cooling vat before it is sent to a fermentation vessel where yeast is introduced to turn the wort into beer. In the final stage the beer is filtered and carbonated before it is bottled and packaged.



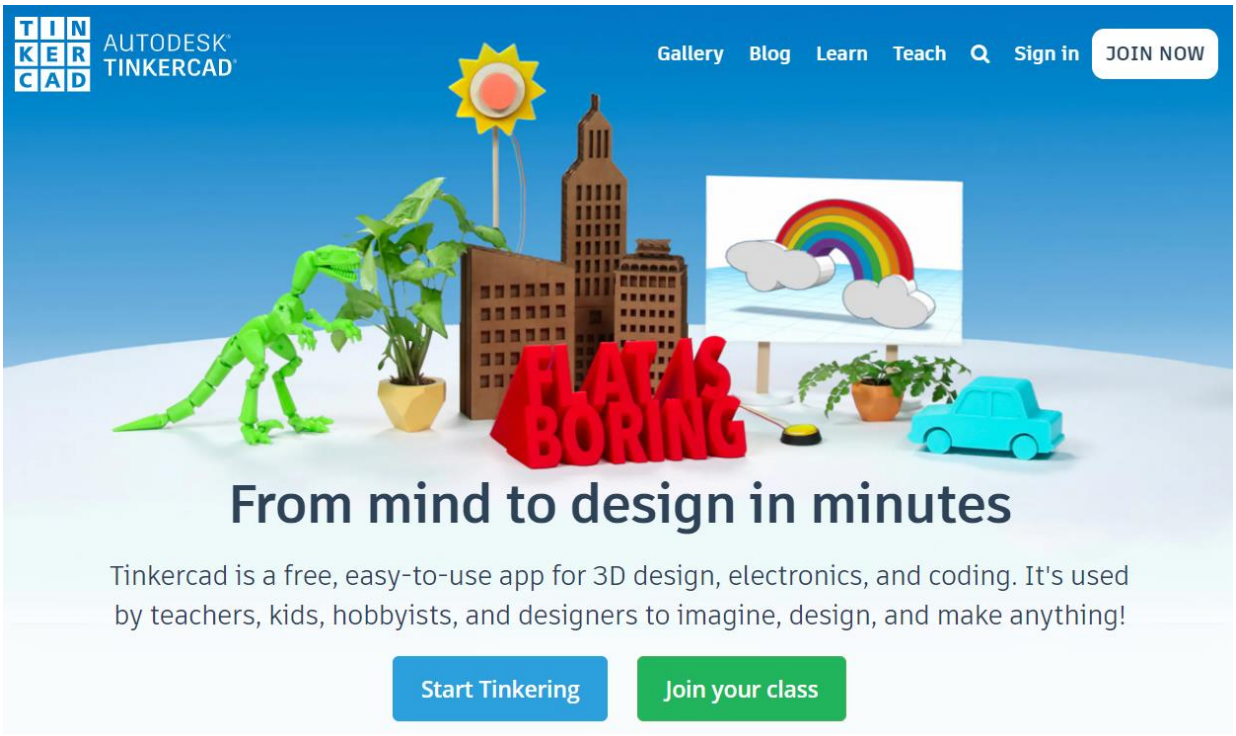
Activity 1. Getting Started

You will use Tinkercad to complete each of these exercises. Tinkercad is a free, online software tool that allows you to create and program Arduino projects virtually without the need for physical equipment.

To sign up for a free Tinkercad account, visit the following website: http://www.tinkercad.com.



Note: You will need to click on Circuits on the left-hand column in order to access the Tinkercad Circuits function of the website.



Tinkercad Account Information

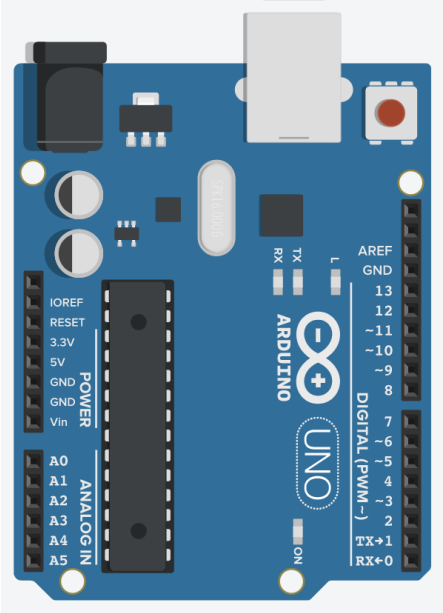
Username: _____

Password*: _____

Keep this document in a safe place

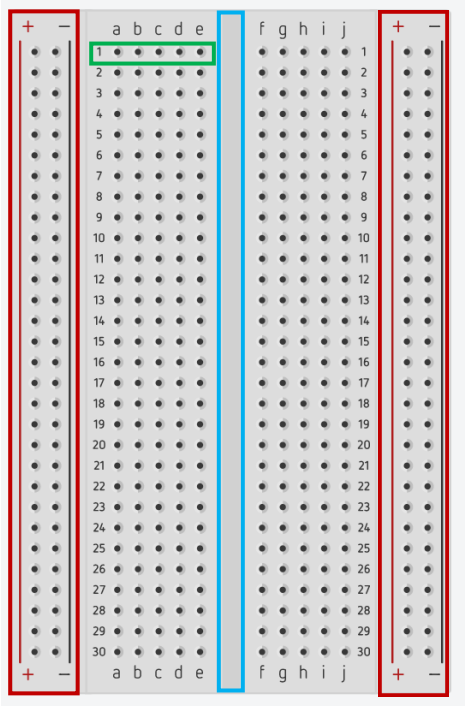
Activity 2. Brief Overview of Required Components

Each project will provide a list of components that you will need to add to your Tinkercad Circuit projects. The table below provides a brief description of each component.



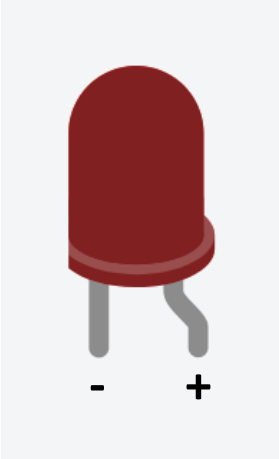
Arduino Uno

- An **Arduino** is an inexpensive, open-source programmable microcontroller designed to allow users to prototype electronics projects quickly and easily.
- **Microcontrollers** are electronic devices that are designed to interface with sensors and actuators. Arduinos can run programs that control these sensors and actuators to communicate with computers and networks.
- The **Arduino Uno** is the most popular board in the Arduino family. It has 14 digital input/output pins and six analog input pins.



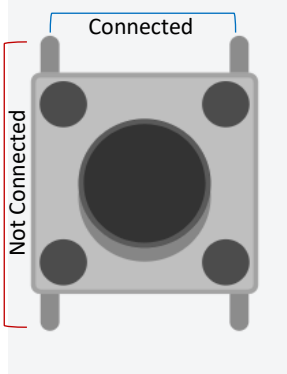
Breadboard

- You will use a solderless **breadboard** to complete the projects. A breadboard is a prototyping platform that allows you to build electronic circuits. This breadboard has two power buses (outlined in red), ten columns, and thirty rows.
- The columns and rows carry electricity through the conductive metal strips underneath the plastic surface and are accessible through the holes in the breadboard. For example, the five holes in row 1 one (outlined in green) are electrically connected.
- The power buses (outlined in red) are used for the power and ground connections. The middle divider (outlined in blue) separate the connection between the two sides of the board.



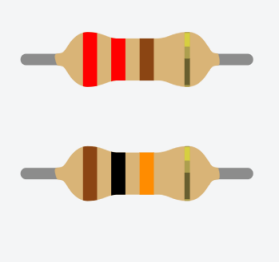
Light Emitting Diode (LED)

- A **LED** is a type of diode that lights up when electricity passes through it. LEDs are polarized components that only allow electricity to flow through them in one direction.
- The longer leg on the LED (depicted with a bent leg in Tinkercad) is called the **anode** and connects to power. The shorter leg is the **cathode** and connects to ground.
- LEDs are often used as indicator lights and use very little power.



Pushbutton Switch

- A **switch** interrupts the flow of electricity and breaks the circuit when open.
- A **pushbutton** is a momentary switch that connect two points in a circuit, and which completes or closes a circuit when pressed.
- The top pins of the switch are connected to each other and the bottom pins are connected to each other.

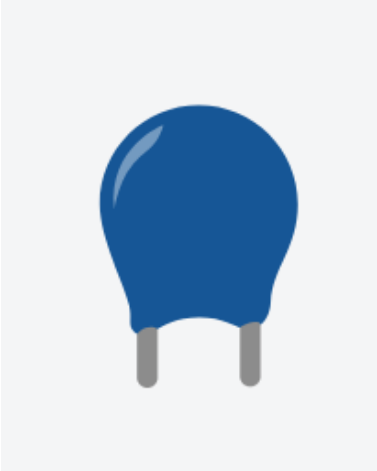


Resistors

- **Resistors** are used to limit the amount of electrical current to protect components, like LEDs. Without using the resistor, the LED could receive too much voltage and quickly burn out.
- Resistance is measured in **ohms** (Ω). Resistors have bands and use a color code to indicate the value of the resistor.
- See the References and Resources at the end of this book for a resistor color code chart.

Micro Servo

- **Servo** motors are rotary actuators that can move to a position accurately and are controlled by sending electrical pulses from the Arduino. These pulses tell the motor what position to move, making these ideal components for electronic applications.
- The servo has three wires coming out. The red is power, the black is ground, and the white is the control line that will receive information from the Arduino.
- A software **library** needs to be imported into your Sketch program in order to work with the servo motor.



Capacitor

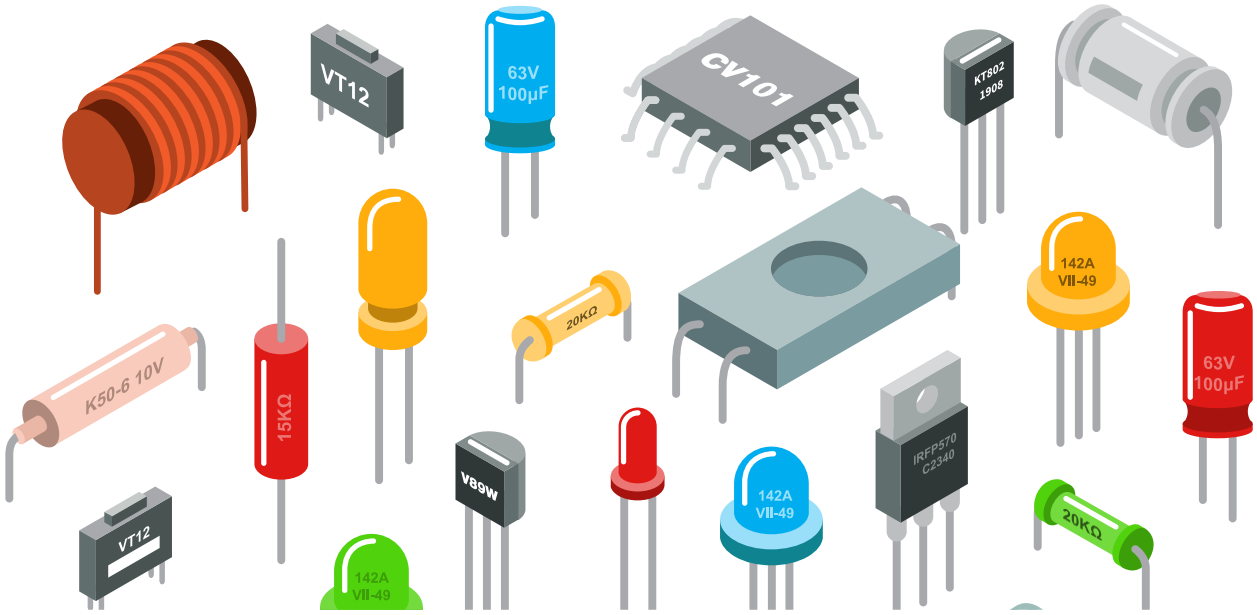
- A **capacitor** is an electrical component that stores energy. When the voltage on a circuit is higher than the energy stored in the capacitor it allows energy to flow in and charge the capacitor. Energy is released from the capacitor when the circuit's voltage is lower.
- Capacitors are often used across the power and ground of a motor or sensors as a way to smooth any changes in voltage.
- The servo motor draws more voltage when it starts to move which causes a dip in voltage on the board. The capacitor smooths out this fluctuation in voltage.

Ultrasonic Distance Sensor (Parallax Ping)

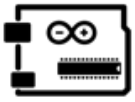
- The **Ultrasonic Distance Sensor** is an input sensor that measures the non-contact distance between moving or stationary items.
- The sensor works by transmitting a non-audible ultrasonic ping and providing an output pulse that corresponds to the time it takes for the ping echo to return to the sensor. A distance to the target is then calculated by measuring the time it takes for the echo to return to the sensor.
- This component has three connections: ground (GND), Power (5V), and an I/O pin (SIG).

Project 1: Using a Pushbutton Switch to Control a LED

Instructions: In this project you will learn some of the fundamentals to building circuits in Tinkercad. This project will allow you to control a LED light using only a pushbutton switch. There is no computer code required for this project.



Project 1 Components and Wiring



Make It

For this project, you will need the following components.

- Arduino Uno
- Small Breadboard
- 1 x Red LED
- 1 x Pushbutton
- 1 x 220 Ω Resistor
- Jumper Wires

Input / Output Devices

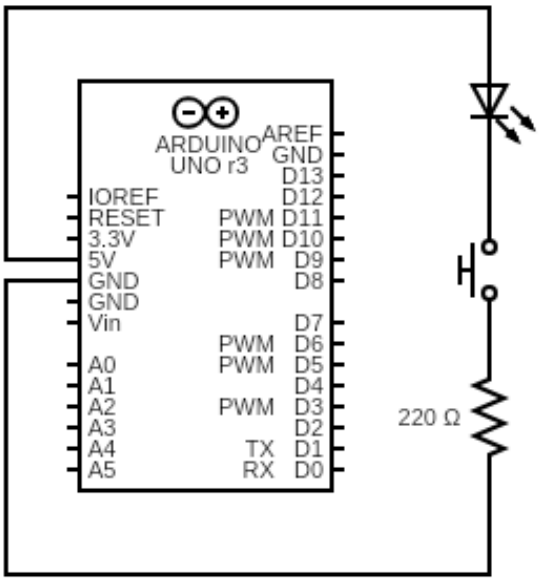
Input: Pushbutton
Output: LED

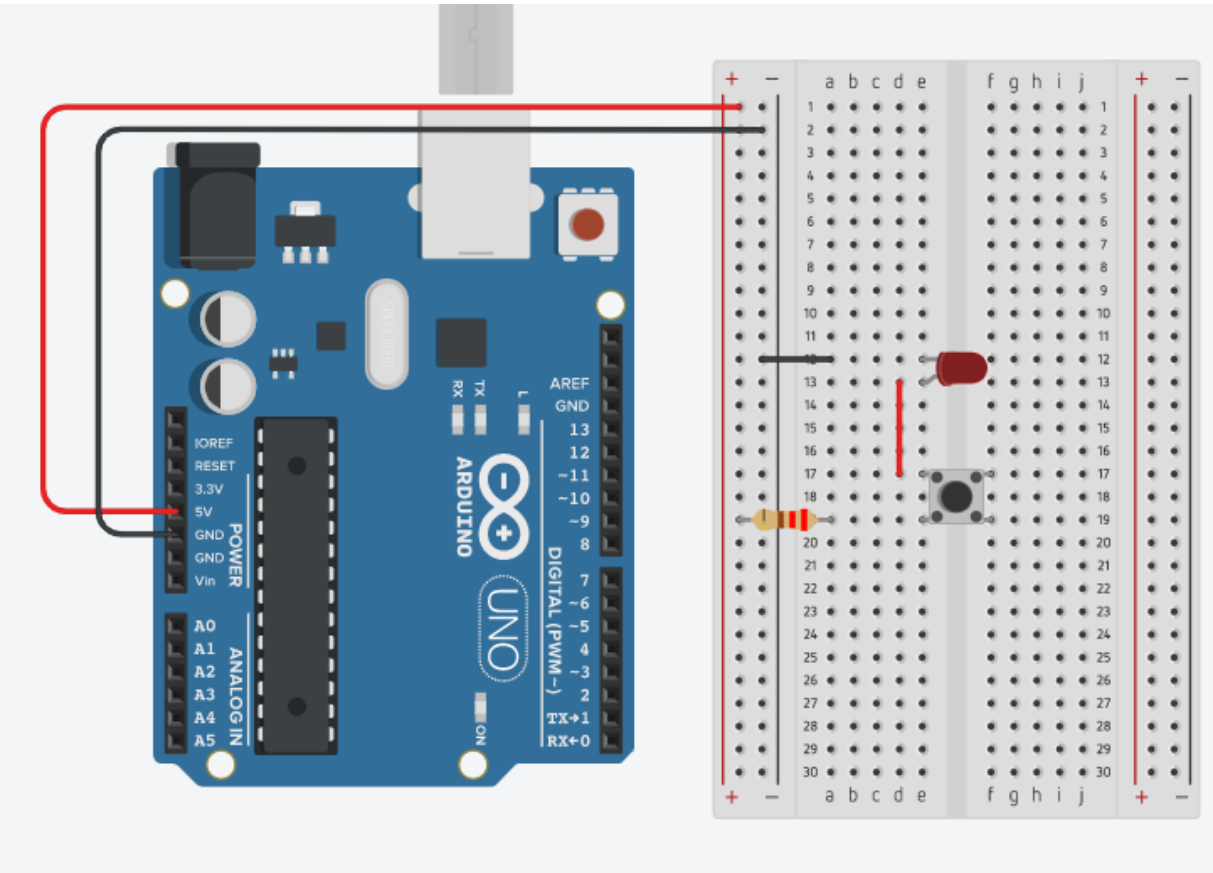
Helpful Wiring Notes:

- The placement of the anode and cathode legs of the LED is important.
- Note the orientation of button on the breadboard.
- The button must be held down to initiate blinking.

Wire It

Use the schematic and circuit illustration below to wire this project in Tinkercad.





Test It

Once the wiring has been completed, you are ready to start the simulation. Click on the simulation button to start it and then click and hold the pushbutton. Did the light go on? If not, check your wiring and resistor value.

How did it go?

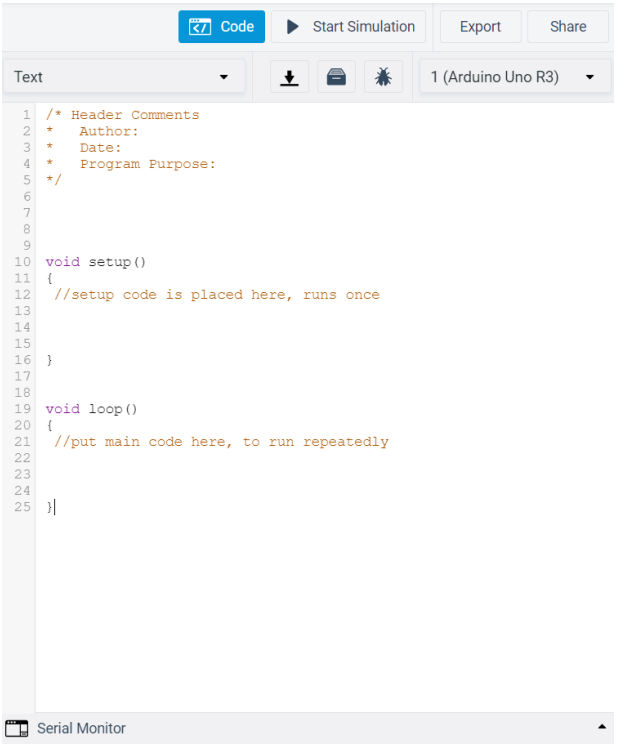
If all went well, congratulations, you now know how to create a circuit that allows you to turn a light on and off using a pushbutton switch. If it did not work, refer to the circuit illustration to ensure you have everything wired correctly.

Notes

LEDs are limited in the amount of power they can receive. In this circuit, the 220 Ω resistor is used to prevent the LED from being overpowered and burning out.

Activity 3. Brief Introduction to Arduino Programming

Many of the Arduino projects will require a basic knowledge of programming principles. For these Arduino projects you will use a simplified version of C++ which is maintained by Arduino.cc.



Using the Tinkercad Circuits Simulator, you can start Arduino programming projects by using either code blocks, a text editor, or both once a programmable component is added to the workspace. For the following projects, you will type your code into the text editor. The Tinkercad Circuit Simulator has a debugger, a serial monitor to view specified output, and an ability to download your code.

An Arduino program is called a **Sketch** and is divided into three main parts: functions, values (variables and constants), and structure. Each will part is briefly described below.

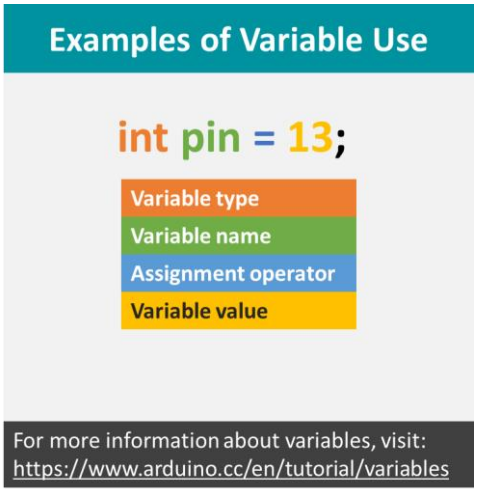
Functions

A **function** is a block of organized, reusable code that is used to perform a single action. All Sketches have 2 special functions: `setup()` and `loop()`.

- The **`setup()` function** is called once when the Sketch starts, when the Arduino board is reset, or after each powerup. This function is used to initialize variables, pin modes, and initializing libraries.
- The **`loop()` function** is used to actively control the Arduino board because it loops consecutively and allows your program to change and respond.

Values – Variables and Constants

Variables are used to store information that can be referenced and manipulated in a computer program. They are stored in the Arduino's memory so that you can keep track of what is happening. Variables must be declared prior to using them. Initialization, although helpful, is not required. Variables and constants have a name, value, and type. Constants use the `const` keyword when they are declared.



- **Variables** values can change when a program is run depending on the instructions of the program. Variables can have global or local scope. Global variables can be used by every function in a program while local scope limits the access of a variable to one function.
    - Example: `int buttonState = 0;`

- **Constants** values do not change while a program runs, they are 'read-only'. Constants can help make programs easier to read and can be used for setting up digital pins.
    - Example: `int const buttonPin = 2;`

Use Good Naming Conventions for Variables and Constants

When naming variables and constants, be sure to assign a name that is accurately descriptive and understandable to another reader. Names of variables and constants are also case sensitive.

Structure

The basic structure of an Arduino sketch consists of two special functions, `setup()` and `loop()`. The flow of the program is top-to-bottom execution, so it is important that you create and follow a consistent structure each time you create a Sketch.

Sample Program Structure to Use

- Header comment: author, date, function of program
- Include libraries (preprocessor commands)
- Declare global variables
- setup() – initialize code here
- loop() – repeated forever after setup() code is run
- Declare functions

An example of the Arduino Sketch structure is displayed on the right.

```
/*
*Title:    Blink without Delay
*Function: Turns on and off a LED
*Author:   Your Name
*Created:  The Date
*/

int const lightOn = 12;  //declare digital pin

void setup()
{
  //setup code - designate pin 12 as output
  pinMode(lightOn, OUTPUT);

}

void loop() {
   //loop runs repeatedly calling blinkOnce
function blinkOnce(1000);
}


void blinkOnce(int delayTime) {
  digitalWrite(led, HIGH); //turn the LED (HIGH)
  delay(delayTime); //wait for a second
  digitalWrite(led, LOW);  //turn the LED off
  delay(delayTime); //wait for a second
}
```

Adding Comments to Your Code

Comments are statements that are used in a sketch to help others understand the purpose of your code. It also makes it easier to debug and maintain code and can also help remind you of why you coded the sketch the way you did. Comments are ignored by the compiler and do not take any space up in the microcontroller's flash memory, their sole purpose is to help the reader understand the purpose of the sketch.

There are two types of comments you can use in your sketch: single-line and multi-line (block) comments. If your comment requires more than ten words, it is advised to use multi-line comments.

Single line comments: // followed by a space

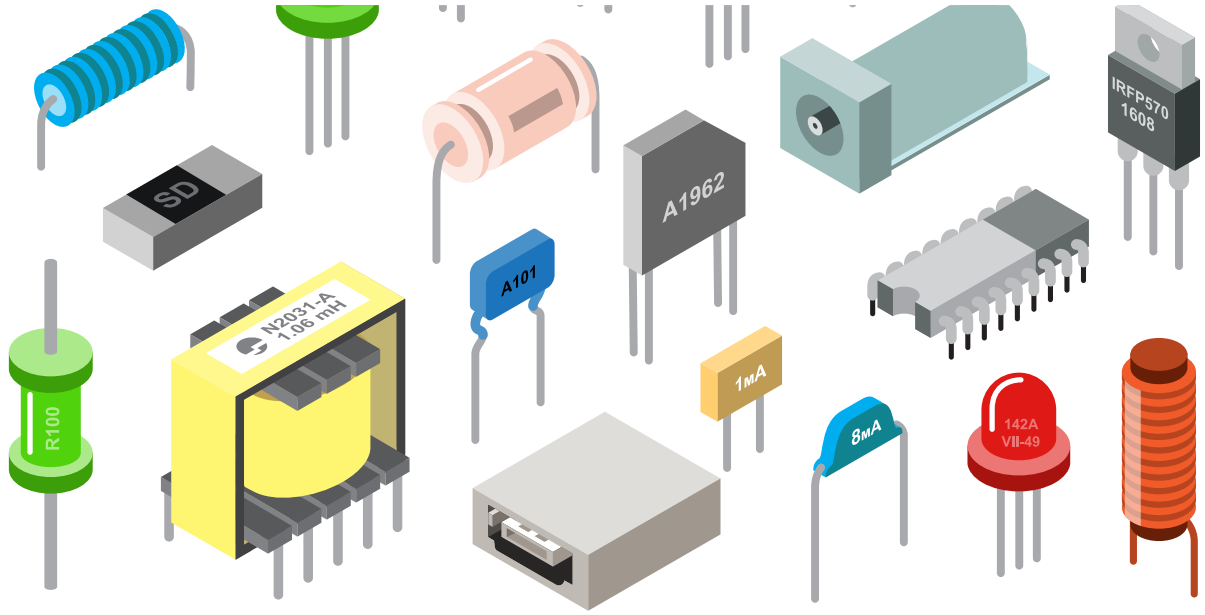
```
// Single line comment
```

Multi-line comments: /* */

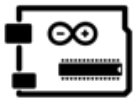
```
/*
* The compiler will assume that everything after the /* symbol
* is a comment until it reaches the */ symbol.
*/
```

Project 2. Using a Push Button Switch and Code to Control a LED

Instructions: In this project you will write a program that accesses the digital input of a pushbutton which will blink a LED light when the button is pressed.



Project 2 Components, Wiring, and Coding



Make It

For this project, you will need the following components.

- Arduino Uno
- Small Breadboard
- 1 x Red LED
- 1 x Pushbutton
- 1 x 220 Ω Resistor
- 1 x 10 kΩ Resistor
- Jumper Wires

Input / Output Devices

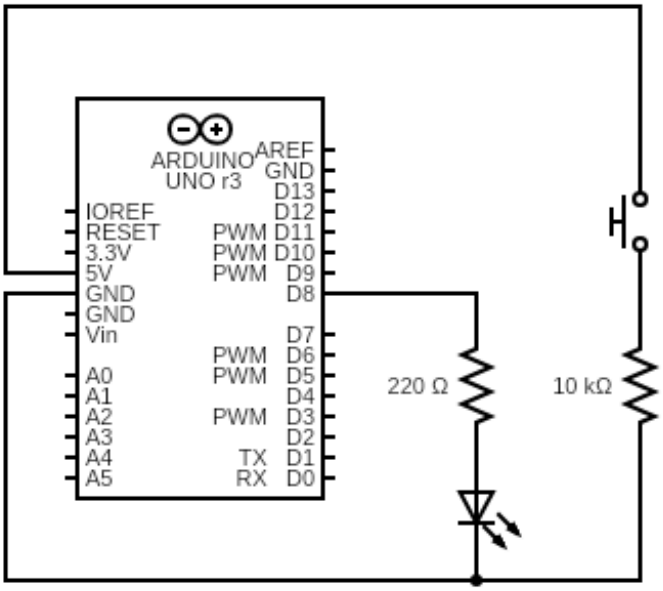
Input: Pushbutton
Output: LED

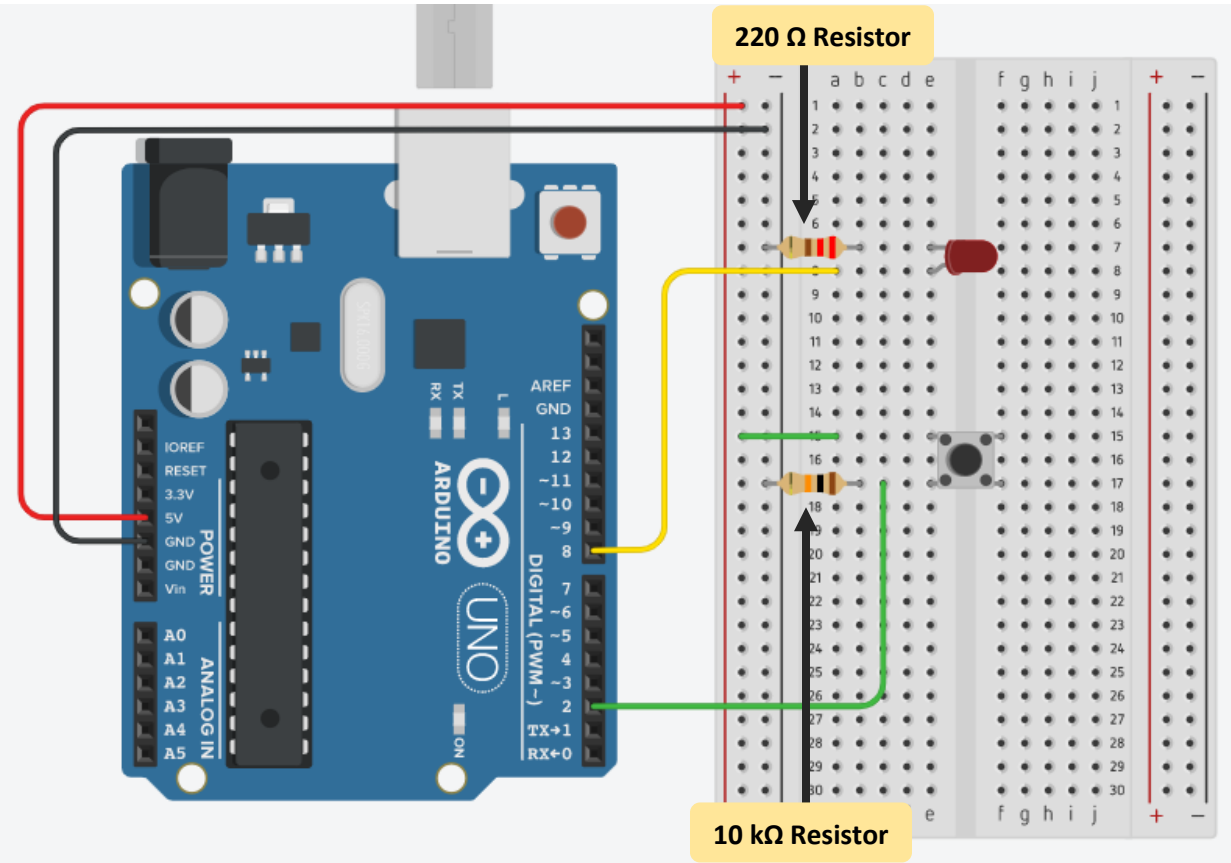
Helpful Wiring Notes:

- The 10 kΩ resistor is connected to the button.
- The 220 Ω resistor is connected to the cathode leg of the LED.

Wire It

Use the schematic and circuit illustration below to wire this project in Tinkercad.





Code It

This simple program will use a push button to turn on and off an LED. While the button remains pressed the LED will blink. **Note:** the line numbers on the left are for reference only and do not need to be included in your code.

```
1  int buttonState = 0;
2
3  int const lightOn = 8;
4  int const buttonPin = 2;
5
6  void setup()
7  {
8    pinMode(lightOn, OUTPUT);
9    pinMode(buttonPin, INPUT);
10 }
11
12 void loop()
13 {
14  buttonState = digitalRead(buttonPin);
15
16   if (buttonState == LOW) {
17
18      digitalWrite(lightOn, LOW);
19  }
20   else {
21     digitalWrite(lightOn, HIGH);
22     delay(1250);
23     digitalWrite(lightOn, LOW);
24     delay(1250);
25   }
26 }
```

Declare and initialize variable for button status

Declare constants for the digital pins

Set the I/O state of the digital pins

This loop will repeat until the program is ended

Determine the current status of the button and store in the buttonState variable
If the button has not been pressed then the LED is turned off

If the button has been pressed then blink the LED on and off at intervals of 1250 ms

Test It

Once the wiring and programming has been completed, you are ready to start the simulation. Start the simulation and click on the push button to see the LED light blink. Did the light blink? If not, check your wiring and code.

How did it go?

Congratulations, you now know how to use digital input to control a LED light and you wrote your first program!

Notes

The 10 kΩ resistor connected to the pushbutton is acting as a pull-down resistor which is used to prevent the input pin from reading a high state (on) when the button is not pressed.

Brewmaster Projects

The brewmaster has asked you to design IoT devices aimed at improving the brewing process. You will be designing and prototyping these devices using Tinkercad. Your job will be to develop new automated processes related to filling the vessels which include:

Brewmaster Project 1. Use a pushbutton switch to open a valve which will allow fluid to enter the vessel.

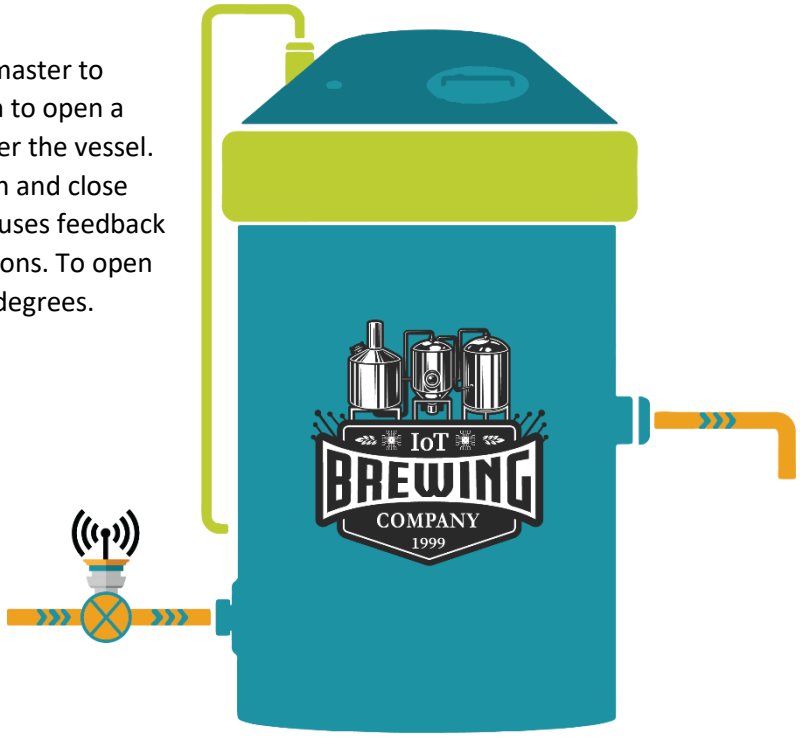
Brewmaster Project 2. Add a light that remains on while the vessel is being filled and will automatically turn off when the valve is closed.

Brewmaster Project 3. Add a sensor to indicate when the vessel is full and when the valve should close automatically.

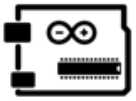
Brewmaster Project 1

Instructions:

You have been asked by the brewmaster to create a device that uses a button to open a valve which will allow fluid to enter the vessel. A servo motor will be used to open and close the valve. A servo motor is an actuator that uses feedback for precise control of angular or linear positions. To open the valve, the servo will need to rotate 180 degrees.



Brewmaster Project 1 Components, Wiring, and Code



Make It

For this project, you will need the following components.

- Arduino Uno
- Small Breadboard
- 1 x Pushbutton
- 1 x 1 kΩ Resistor
- 1 x Micro Servo
- 1 x 100 µF Capacitor
- Jumper Wires

Input / Output Devices

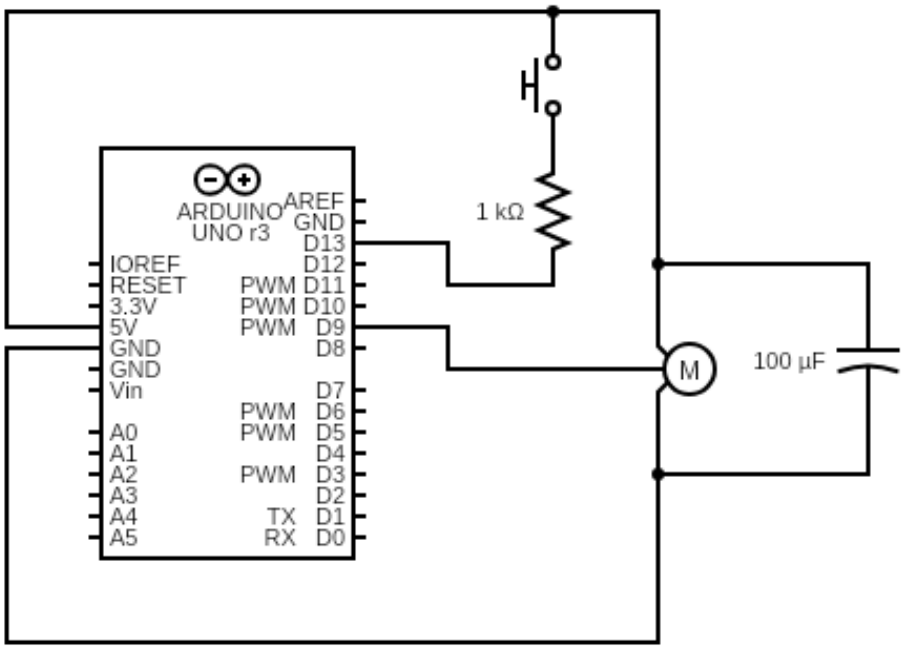
Input: Pushbutton
Output: Micro Servo

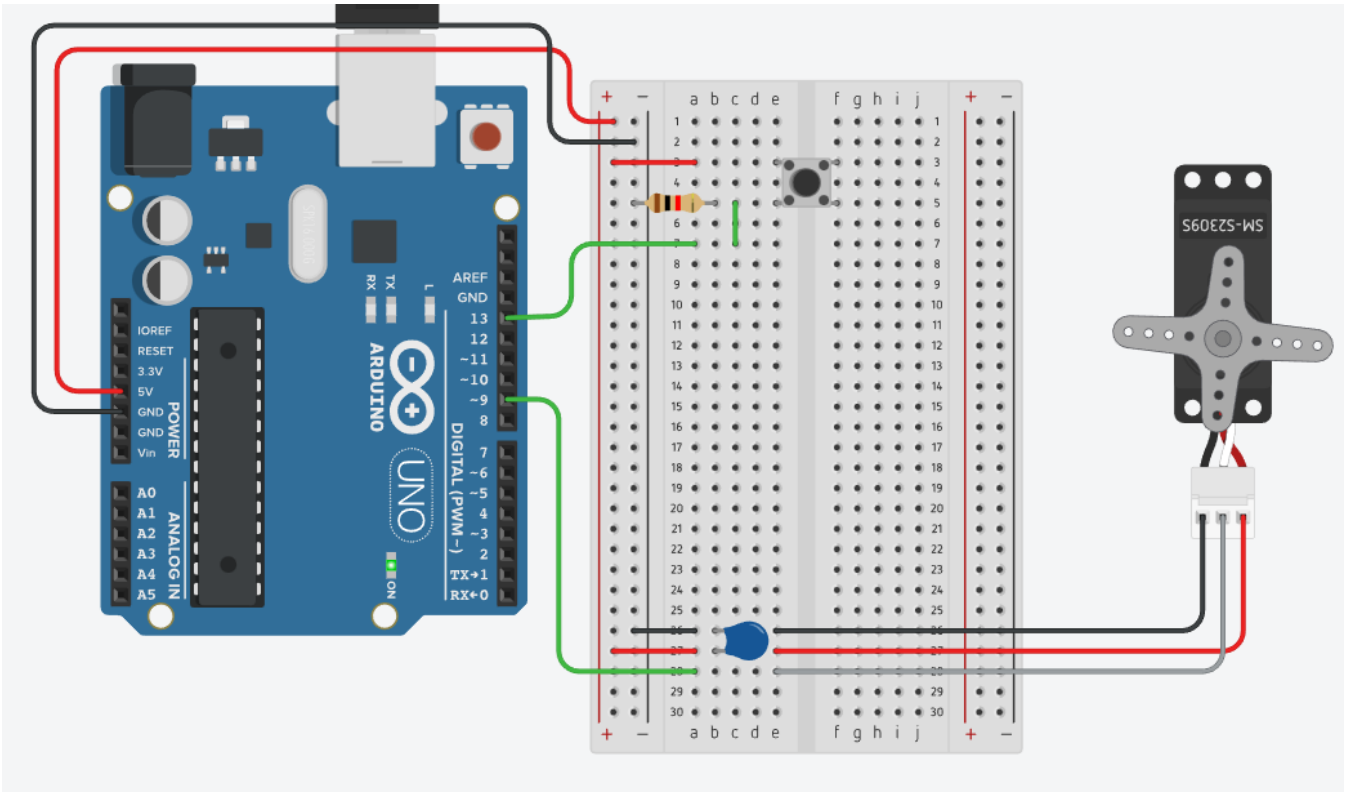
Helpful Wiring Notes:

- The white servo wire is the signal wire, the black wire is ground, and the red wire is for power.

Wire It

Use the schematic and circuit illustration below to wire this project in Tinkercad.





Code It

Below is the code you will need to add to the project in order for it to work. The program will cause the servo to rotate to the open or close position when a button is pressed. Add comments to this code so that its purpose and functions can be clearly understood by other readers.

```
#include <Servo.h>
Servo valveServo;

int const onOffPin = 13;
int const servoPin = 9;

int const valveOpen = 180;
int const valveClose = 0;

bool isOpen = false;
int onOffState = 0;

void setup(){
  valveServo.attach(servoPin);
  Serial.begin(9600);
  pinMode(onOffPin, INPUT);

  valveServo.write(valveClose);
}

void loop(){

  onOffState = digitalRead(onOffPin);

  if (onOffState == HIGH){

    if (isOpen == true){
      valveServo.write(valveClose);
      delay(1500);
      isOpen = false;
    }
    else {
      valveServo.write(valveOpen);
      delay(1500);
      isOpen = true;
    }
  }
}
```

Import the library for the servo
Create the Servo object

Declare constants for the digital pins

Declare constants for the servo position

Declare and initialize initial state variables

Attach the servo to the digital pin, initialize the serial port for communication, and set the I/O status of the digital pin for the button

Initialize the servo starting position

This loop will repeat until the program is ended

Store the current button state

If the button is pressed…

… and the valve is open then close the valve. Delay 1.5 seconds for the servo to operate and update the status of the valve.

… and the valve is closed then open the valve. Delay 1.5 seconds for the servo to operate and update the status of the valve.

Test It

Once the wiring has been completed and the code has been successfully entered you are ready to start the simulation. To open the valve, press the button once and the servo should rotate 180 degrees thus opening the valve to allow fluid into the vessel. If you push the button again the servo should rotate 180 degrees in the opposite direction thus closing the valve.

How did it go?

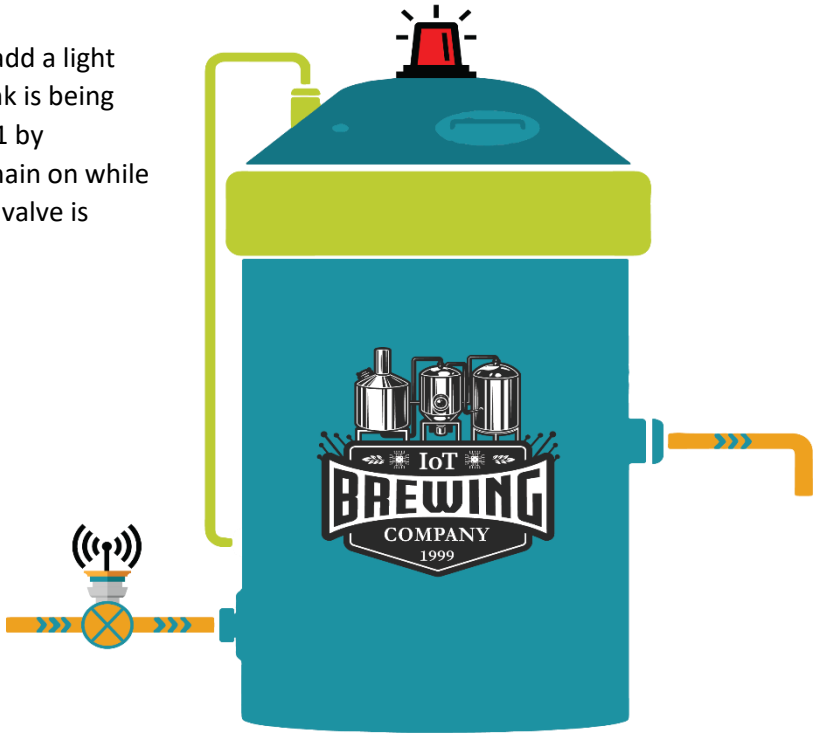
Congratulations, you now know how to open and close a valve using an Arduino and servo motor. Save your work and now you are ready to proceed to the second project.

Notes

Brewmaster Project 2

Instructions:

The brewmaster asked you to add a light that will indicate when the tank is being filled. You will add to Project 1 by introducing a light that will remain on while the vessel fills and will turn off when the valve is closed.



Brewmaster Project 2 Components, Wiring, and Code

Make It

Input / Output Devices

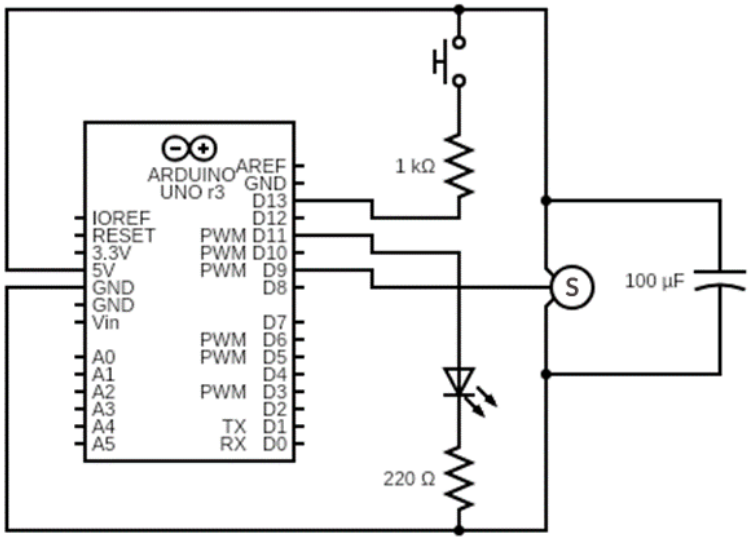
Input: Pushbutton
Output: LED, Micro Servo

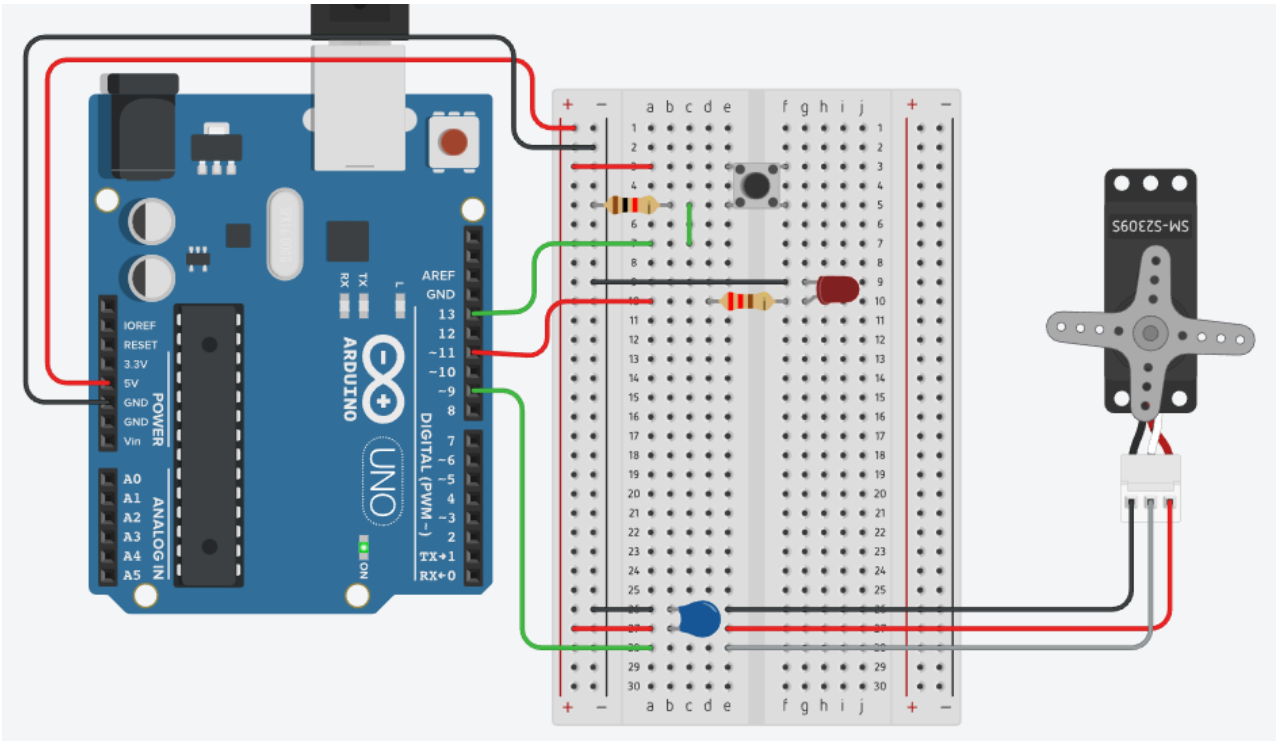
Add the following components to your circuit

- 1 x Red LED
- 1 x 220 Ω resistor
- Jumper wires

Wire It

Use the schematic and circuit illustration below to wire to add the additional components to your breadboard.





Code It

Add the code highlighted **bold** below. The code additions will turn the LED on when the valve is open and turn the LED off when the valve is closed.

```
1  #include <Servo.h>
2  Servo valveServo;
3
4  int const onOffPin = 13;
5  int const servoPin = 9;
6  int const LEDPin = 11;
7
8  int const valveOpen = 180;
9  int const valveClose = 0;
10 bool isOpen = false;
11
12 int onOffState = 0;
13 int previousOnOffState = 0;
14
15 void setup(){
16   valveServo.attach(servoPin);
17   Serial.begin(9600);
18   pinMode(onOffPin, INPUT);
19   pinMode(LEDPin, OUTPUT);
20   valveServo.write(valveClose);
21 }
22
23 void loop(){
24
25   onOffState = digitalRead(onOffPin);
26
27   if (onOffState == HIGH){
28
29     if (isOpen == true){
30       valveServo.write(valveClose);
31       delay(1500);
32       isOpen = false;
33       digitalWrite(LEDPin, LOW);
34     }
35     else {
36       valveServo.write(valveOpen);
37       digitalWrite(LEDPin, HIGH);
38       delay(1500);
39       isOpen = true;
40     }
41   }
42 }
```

Add a constant declaring the digital pin to be used with the LED

Set the I/O status of the LED pin

When the valve is closed turn the LED off

When the valve is opened turn the LED on

Test It

Once the additional components and code have been added, test to see whether the light remains on while the valve is open and turns off when the valve is closed.

How did it go?

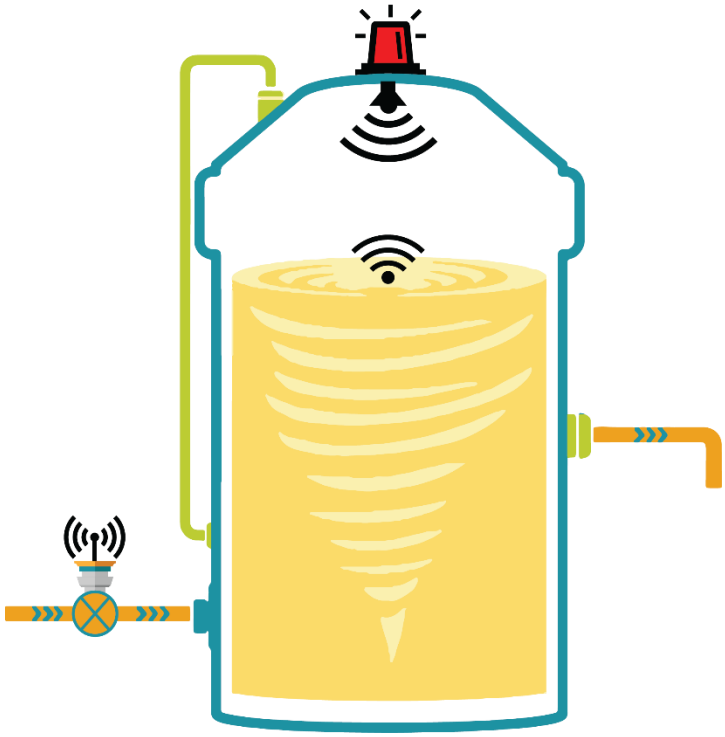
Congratulations, you now added a light indicator that will let you know whether the valve is open and closed. Save your work and now you are ready to proceed to Project 3.

Notes

Brewmaster Project 3

Instructions:

The brewmaster wants to automatically turn the valve off when fluid in the vessel reaches a certain level. You can do this by adding a proximity sensor to read the fluid level in the tank. The sensor will be attached to the underside of the lid inside the vessel. In this way, the sensor will read the level of the fluid as the tank fills. Once the fluid reaches 100 cm from lid the valve should close.



Brewmaster Project 3 Components, Wiring, and Code

Make It

Add the following component to your circuit:

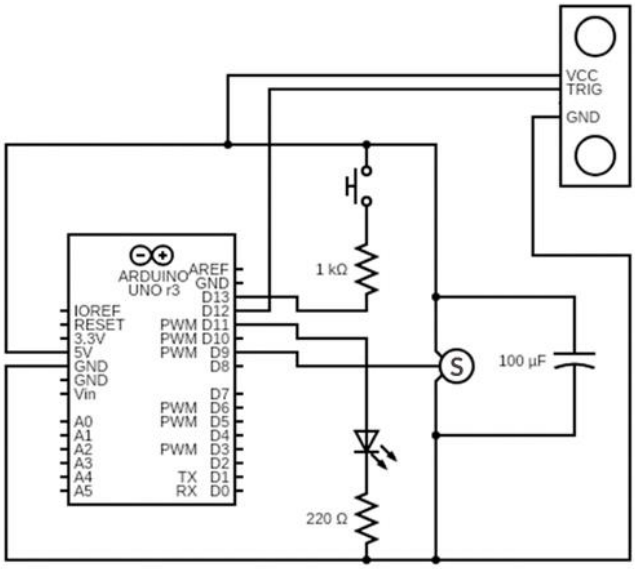
- 1 x Ultrasonic Distance Sensor (Parallax Ping)
- Jumper wires

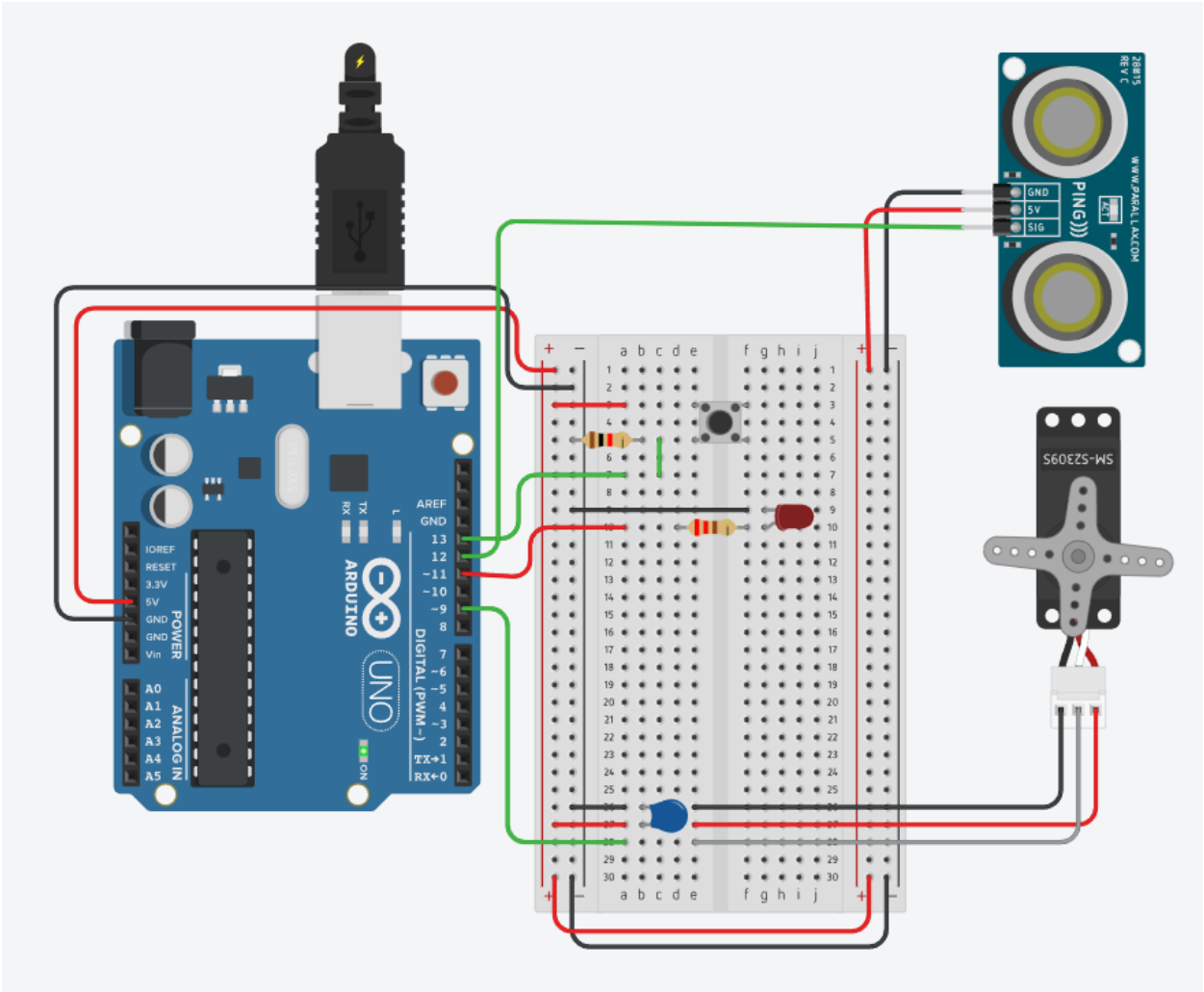
Input / Output Devices

Input: Ultrasonic distance sensor, pushbutton
Output: LED, Micro Servo

Wire It

Use the schematic and circuit illustration below to wire to add the additional components to your breadboard.


VCC
TRIG
GND
1 kΩ
100 µF
220 Ω
ARDUINO
UNO r3
AREF
GND
D13
D12
IOREF
RESET
3.3V
5V
GND
GND
Vin
PWM D11
PWM D10
PWM D9
D8
D7
PWM D6
PWM D5
D4
PWM D3
D2
TX D1
RX D0
A0
A1
A2
A3
A4
A5


PING)))
GND
5V
SIG
SM-S2309S
ARDUINO
UNO
POWER
ANALOG IN
DIGITAL (PWM~)

Code It

Add the highlighted code. The code additions will introduce a PING sensor that will sense the fluid level as the tank fills. When the fluid reaches a predetermined level, the valve will automatically close and the tank will stop filling.

```
1  #include <Servo.h>
2  Servo valveServo;
3
4  int const onOffPin = 13;
5  int const servoPin = 9;
6  int const LEDPin = 11;
7  int const pingPin = 12;
8
9  int const valveOpen = 180;
10 int const valveClose = 0;
11 bool isOpen = false;
12 int const fluidLevel = 100;
13
14 int onOffState = 0;
15 int previousOnOffState = 0;
16
17 long duration, cm;
18
19 void setup(){
20   valveServo.attach(servoPin);
21   Serial.begin(9600);
22   pinMode(onOffPin, INPUT);
23   pinMode(LEDPin, OUTPUT);
24
25   valveServo.write(valveClose);
26 }
27
28 void loop(){
29
30   onOffState = digitalRead(onOffPin);
31
32   pinMode(pingPin, OUTPUT);
33
34   digitalWrite(pingPin, LOW);
35   delay(2);
36
37   digitalWrite(pingPin, HIGH);
38   delay(5);
39
40   digitalWrite(pingPin, LOW);
41
```

Add a constant declaring the digital pin to be used with the distance sensor

Declare and initialize the maximum fluid level in the tank

Declare variables for the duration of the ping and the distance to the sensor

Set the status of the sensor pin to output to begin the triggering event

Send the sensor a LOW pulse for 2 ms to ensure a good HIGH pulse

The sensor is triggered by sending a HIGH pulse for 5 ms

Send the sensor a LOW pulse

```
42    pinMode(pingPin, INPUT);
43    duration = pulseIn(pingPin, HIGH);
44
45    cm = msecToCm(duration);
46
47    if (onOffState == HIGH){
48
49      if (isOpen == true){
50        valveServo.write(valveClose);
51        delay(1500);
52        isOpen = false;
53
54        digitalWrite(LEDPin, LOW);
55      }
56
57      else {
58        valveServo.write(valveOpen);
59        digitalWrite(LEDPin, HIGH);
60        delay(1500);
61        isOpen = true;
62      }
63    }
64
65    if(cm <= fluidLevel) {
66      valveServo.write(valveClose);
67      digitalWrite(LEDPin, LOW);
68      delay(1500);
69      isOpen = false;
70    }
71
72  }
73
74  long msecToCm(long microseconds){
75    return microseconds/29/2;
76  }
```

Change the status of the sensor read the echo
Determine the duration between pulse and echo

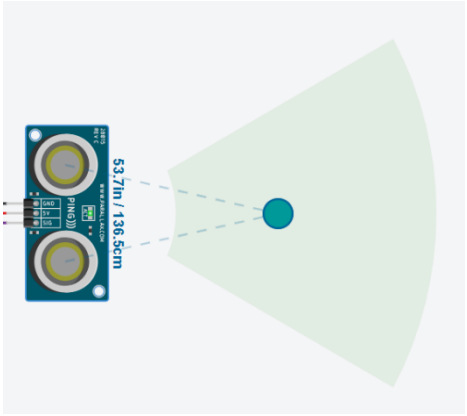
Convert the duration to a distance

If the distance is at the desired fluid level close the valve and turn off the LED

Function to convert the ping duration into distance: The speed of sound is 340 m/s or 29 ms/cm. The ping travels out and back, so to find the distance to the object divide by 2

Test It

Once the additional components and code have been added, test to see whether the distance sensor closes the valve once the fluid is 100 cm or less away from the vessel lid. Start the simulation to test whether your project works. The proximity sensor requires that you click on the sensor and then drag the ball close to the sensor to simulate the fluid filling the tank. Drag the ball until it reads 100cm or less to see the valve close and the light turn off.

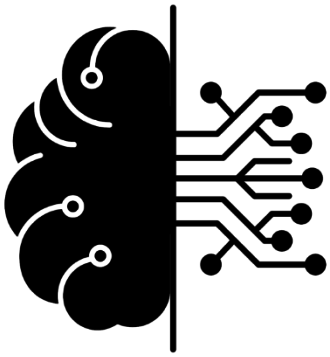


How did it go?

Congratulations, you have now used a proximity sensor to automate the tank filling process.

Notes

Looking for Additional Challenges?



Project 1: Use the Serial Monitor to view the output to determine whether the servo motor is open or closed.

Project 2: Make the LED light blink as the vessel fills.

Project 3: Use the Serial Monitor to view the fluid level in the vessel as it fills.

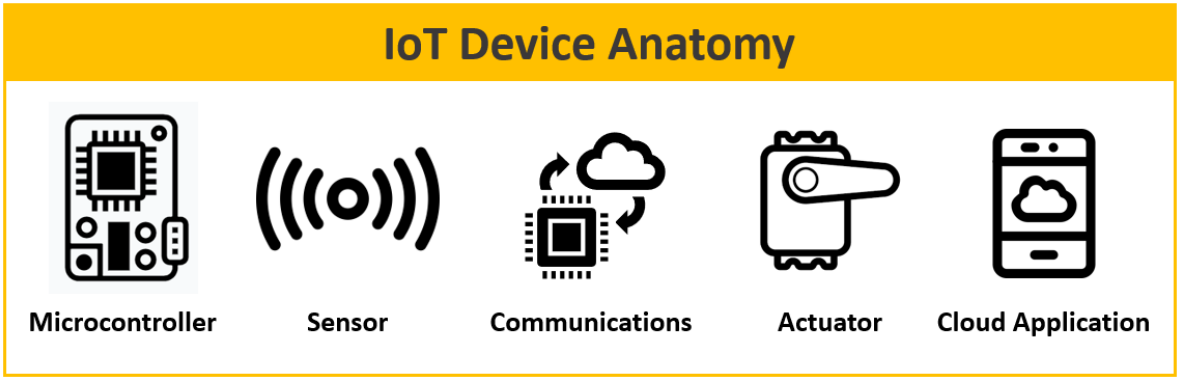
Project 3: Open a valve to empty a vessel when the process has been completed and the fluid is ready to be transferred to the next vessel.

Making the Brewmaster Projects Internet of Things Devices

Tinkercad is limited in its simulation capabilities and does not provide any devices to simulate an IoT device. However, there are only minor changes and adjustments that need to be undertaken to create fully IoT devices from the Brewmaster Projects. To make the Brewmaster Projects true Internet of Things devices requires the following modifications:

1. Switch the Arduino Uno microcontroller with a board capable of communicating with a network such as the Arduino Nano 33, Arduino MKR1000, Arduino MKR WiFi1010, or similar. In some cases, where the power of the IoT board is limited to 3.3 V, slight alteration to the circuit may be necessary such as changing the resistor values.
2. Utilize an IoT cloud service, such as the Arduino IoT Cloud, to create widgets or apps to interface with the IoT device. The button and LED used in the Brewmaster Projects may be duplicated on the IoT cloud service so that activations can be performed, and alerts can be visualized, via the service page or app.
3. Add programmatic elements to initiate communication with a webpage or IoT cloud service.

For more information visit the Arduino IoT Cloud homepage: https://www.arduino.cc/en/IoT/HomePage



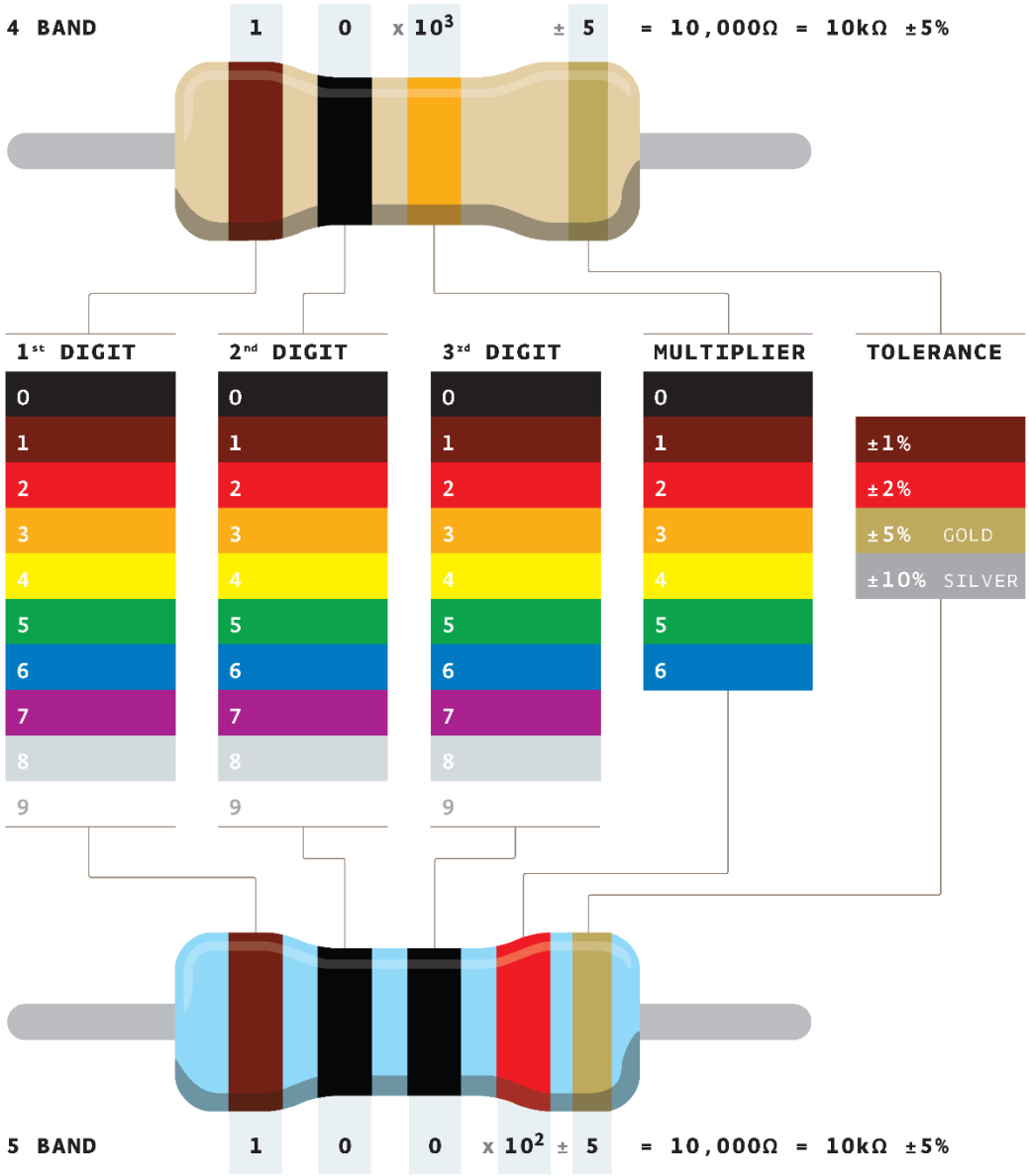
Glossary of Terms

Actuator	A type of component that changes energy into motion. Motors are a type of electrical actuator.
Analog	Something that can continuously vary over time.
Anode	The positive end of a diode (remember that an LED is a type of diode).
Arduino	An open-source programmable microcontroller designed to allow users to prototype electronics projects quickly and easily.
Boolean	A datatype that indicates something binary, such as *on* or *off*, *1* or *0*.
Breadboard	A prototyping platform that allows you to build electronic circuits.
Capacitor	A component that can hold an electrical charge.
Cathode	The negative end of a diode.
Comments	Statements that are used to help others understand the purpose of your code.
Constants	A named identifier that cannot change its value in a program.
Circuit	A circular path from a power supply, through a load, and then back again to the other end of the power supply. Current flows in a circuit only if it is closed, that is, if the outgoing and return path are both uninterrupted (or closed). If either path is interrupted (or open) then current will not flow through the circuit.
Digital	A system that deals with discrete values (typically 1s & 0s).
Function	A block of code that executes a specific task.
Internet of Things	A network of connected devices enabling them to communicate over the internet.
Library	A software extension of the Arduino API that expands the functionality of a program.
Light emitting diode (LED)	A type of diode that lights up when electricity passes through it. LEDs are polarized components that only allow electricity to flow through them in one direction.
Microcontrollers	The brains of the Arduino, this is a small computer that you will program to listen for, process, and display information.

Ohms (Ω)	Unit of measurement of resistance. Represented by the omega symbol.
Polarized	The leads of polarized components (e.g. LEDs or capacitors) have different functions, and thus must be connected the right way. Polarized components connected the wrong way might not work, might be damaged, or might damage other parts of your circuit. Non-polarized components (e.g. resistors) can be connected either way.
Prototyping	An initial stage of product development in which a working model is constructed and tested before the final product is manufactured.
Pull-down Resistor	A pull-down resistor (or pull-up resistor) is a resistor used to ensure a known state for a digital signal. It is typically used with buttons and switches to ensure an errant high state is not read when the button or switch is not pressed.
Resistor	A measure of how efficiently a material will conduct electricity
Sensor	A component that measures one form of energy (like light or heat or mechanical energy) and converts it to voltage or current.
Servo	Servo motors are rotary actuators that can move to a position accurately and are controlled by sending electrical pulses from the Arduino.
Sketch	The term given to programs written in the Arduino IDE.
Switch	A component that can open or close an electrical circuit.
Ultrasonic Distance Sensor	The Ultrasonic Distance Sensor is an input sensor that measures the non-contact distance between moving or stationary items.
Valve	A device for controlling the passage of fluid or air.
Variables	A datatype that stores values which are likely to change as your program runs. A variable's type depends on the type of information you want to store, and the maximum size of the information.

* Many of the terms included in the Glossary were taken from the Arduino Glossary (https://www.arduino.cc/glossary/en/)

Resistor Color Code Chart



Credit: Adim Kassn / CC BY-SA
https://commons.wikimedia.org/wiki/File:Resistor_color_code.png

Additional Arduino Resources

- Arduino.cc - https://www.arduino.cc/
- What is Arduino? - https://www.arduino.cc/en/Guide/Introduction
- Digital Pins - https://www.arduino.cc/en/Tutorial/DigitalPins
- Arduino IoT Cloud - https://www.arduino.cc/en/IoT/HomePage