Arduino Intro to Servo Motors

By: Matthew Jourden

Brighton High School

Brighton, MI

Motor Basics

There are three types of motors that can be used in an Arduino Circuit.

Stepper: Stepper motors, due to their high pole (50 to 100) count, offer precision drive control for motion control applications. They have a high torque at low speeds, and they're also relatively inexpensive and widely available.

Stepper motors have limitations though. At high-speeds, they lose nearly all of their torque, sometimes up to 80%. They produce high vibrations levels and are prone to resonance issues. Stepper motors also produce high amounts of heat, which can be an issue in certain applications.

Each motor will have three wires On each motor there is a circuit board that needs power, the Voltage and Ground wires complete this circuit. The Signal Wire allows communication between the Arduino and the motor.

Wires

- V: Voltage, Red Wire
- G: Ground, Brown or Orange Wire
- S: Signal, White Wire

Servo: The main benefit of servo motors is they provide high levels of torque at high speed – something stepper motors can't do. They also operate at 80 – 90% efficiency. Servo motors can work in AC or DC drive, and do not suffer from vibration or resonance issues.

Servo motors have many advantages including the programming, tracking and speed control of current angles of the motor.

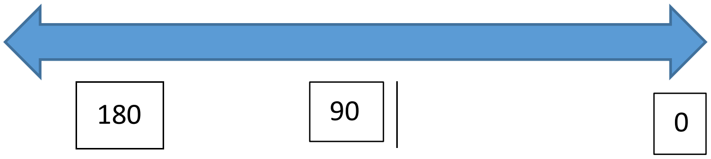
Two Types

i. **360 Degree Motors:** Motor Pans 0-180 Degrees. User will assign a specific Angle 0-180 that will rotate the Servo Motor to that degree a max of 180 degrees (i.e. myservo.write (45); will rotate the servo motor to 45th degree.)

   NOTE: turn servo motor from 0 to 180 then from 180 to 0 degrees gets 360 degrees

ii. **360 Degree Continuous:** allows the user to rotate full 360 degree circle. Programming the servo will still use values of 0 and 180 as start and end points to determine the number of rotations.

   1. 0 = Full Speed Clockwise
   2. 90 = Neutral position
   3. 180 = Full Speed Counter Clockwise



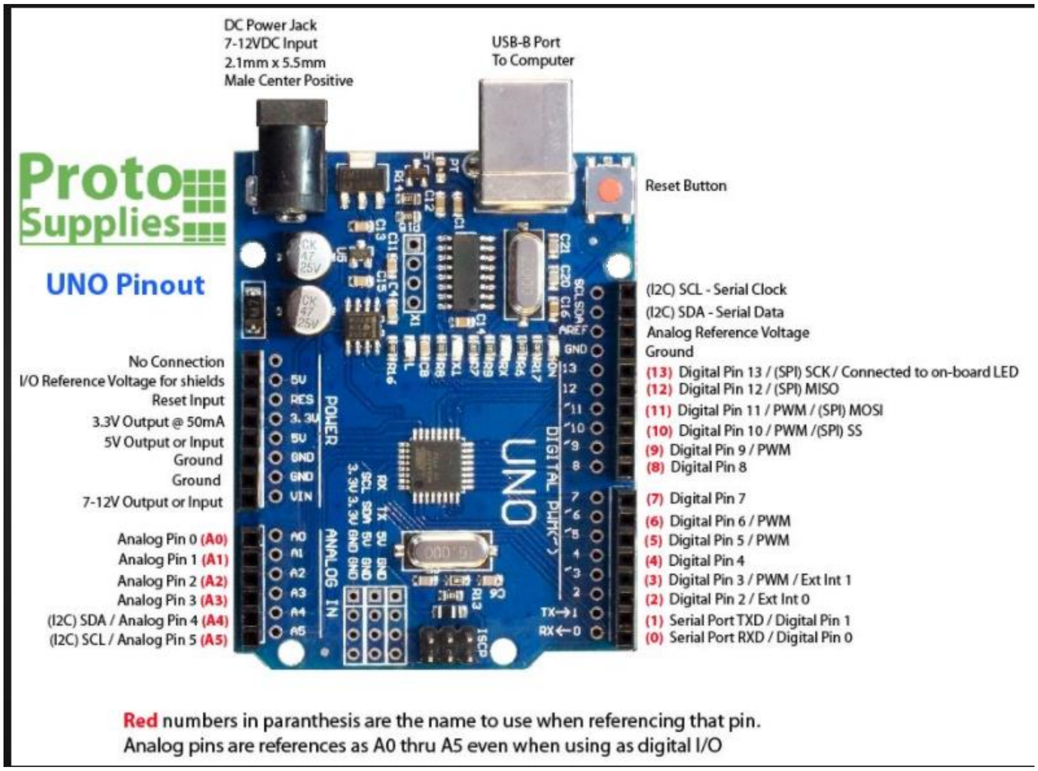
Each motor will have three wires. On each motor there is a circuit board that needs power, the Voltage and Ground wires complete this circuit. The Signal Wire allows communication between the Arduino and the motor.

Wires

- V: Voltage, Red Wire
- G: Ground, Brown or Orange Wire
- S: Signal, White Wire

DC Motor: This motor only has a Red (Voltage/Signal) and Ground (Black Wire). This motor has two states On and Off similar to an LED. Speed can be controlled by using potentiometer or a series of resistors.
NOTE: Some DC Motors can have a signal wire. For our use they will not.

Arduino Boards:. Servo Motors will utilize the 5V and Ground ports to power the circuitry in the Servo Motor and Digital Port for the Signal to set position of the Servo Motor. For DC Motors the Red Wire can be placed either in the 5V to have the motor always on or placed in a Digital Port to act to send a Signal: HIGH/LOW for ON/OFF



Red numbers in paranthesis are the name to use when referencing that pin.
Analog pins are references as A0 thru A5 even when using as digital I/O

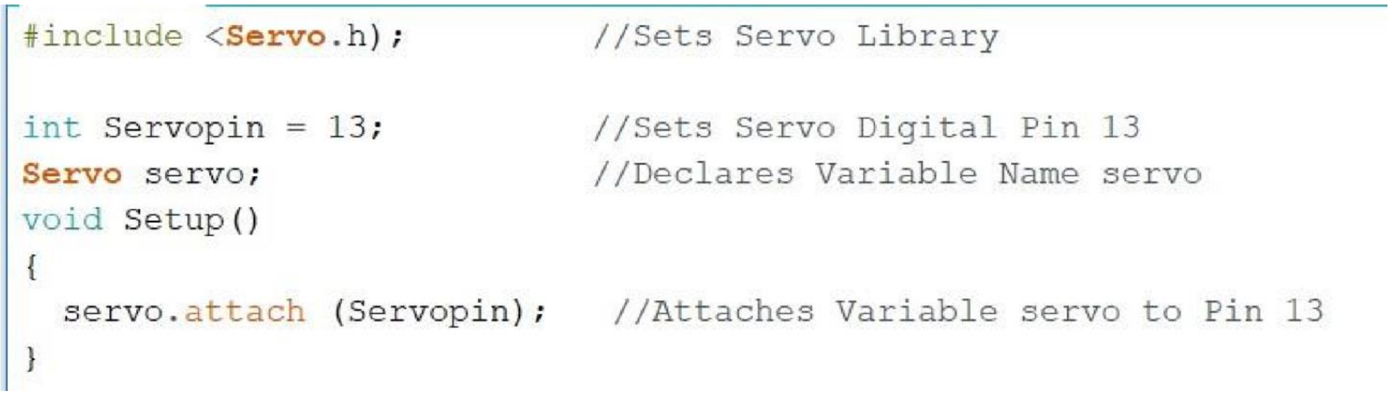
Motor Control Syntax

a. #include <Servo.h> : Library for a the servo motors to send a digital signal to the motor to turn on or off

b. Variable Type Servo : variable that allows the programmer to set a variable name for the motor(s) that are being used

c. “servo name”.attach (“servo pin location”) : Allows user to attach the variable that is declared as a Servo Type to a specific pin location on the Arduino

For Example



```
#include <Servo.h);          //Sets Servo Library

int Servopin = 13;           //Sets Servo Digital Pin 13
Servo servo;                 //Declares Variable Name servo
void Setup()
{
  servo.attach (Servopin);    //Attaches Variable servo to Pin 13
}
```

d. 360 Degree Motors the angle can be programmed from 0 to 180 into the motor and turn that number of degrees. Using an encoder will set the origin (datum) point for the motor. It is important to calibrate these motors, so the motor does not try to over traverse itself and break the gears. NOTE: TinkerCAD will reset the motor back to 0 degree position after each simulation

e. “servo name”.detach (); : Removes power from the Variable Servo

3. Motor Movement: Servo Motors work based on degrees. There is an encoder built into the motor housing that sense how many degrees have turned

   a. 2 Types of Motors

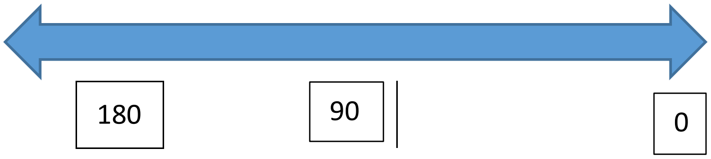
      i **360 Degree Motors:** Motor Pans 0-180 Degrees. User will assign a specific Angle 0-180 that will rotate the Servo Motor to that degree a max of 180 degrees (i.e. myservo.write (45); will rotate the servo motor to 45th degree.)
      NOTE: turn servo motor from 0 to 180 then from 180 to 0 degrees gets 360 degrees

      ii **360 Degree Continuous:** allows the user to rotate full 360 degree circle. Programming the servo will still use values of 0 and 180 as start and end points to determine the number of rotations.

         1 0 = Full Speed Clockwise

         2 90 = Neutral position

         3 180 = Full Speed Counter Clockwise



   b. One can adjust speed and turning based upon delay commands and by attaching and detaching the motor from the Arduino board

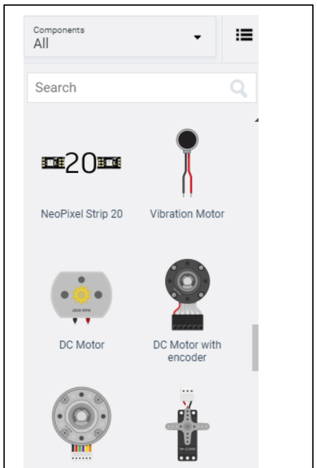
Basic Circuit and Motor Program

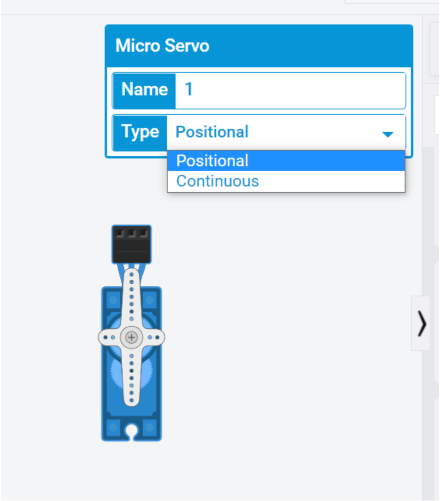
1. Navigate to TinkerCAD > Create the following circuit > Circuit > Create a New Circuit > Rename to Basic Servo Create the following Circuit

NOTE: Change the Components Setting to All (Right Hand Side of the Scree); to see all components

NOTE: Black Wire (Ground) can into any of the three GND Ports

NOTE TInkerCAD has 360 Degree Servo (0-180 Degrees) and 360 Degree Continuous

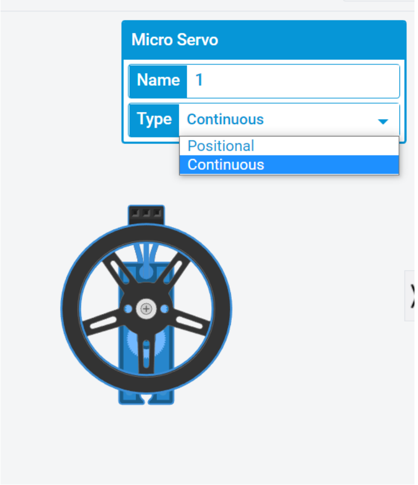

Components All



Micro Servo

Name 1

Type Positional

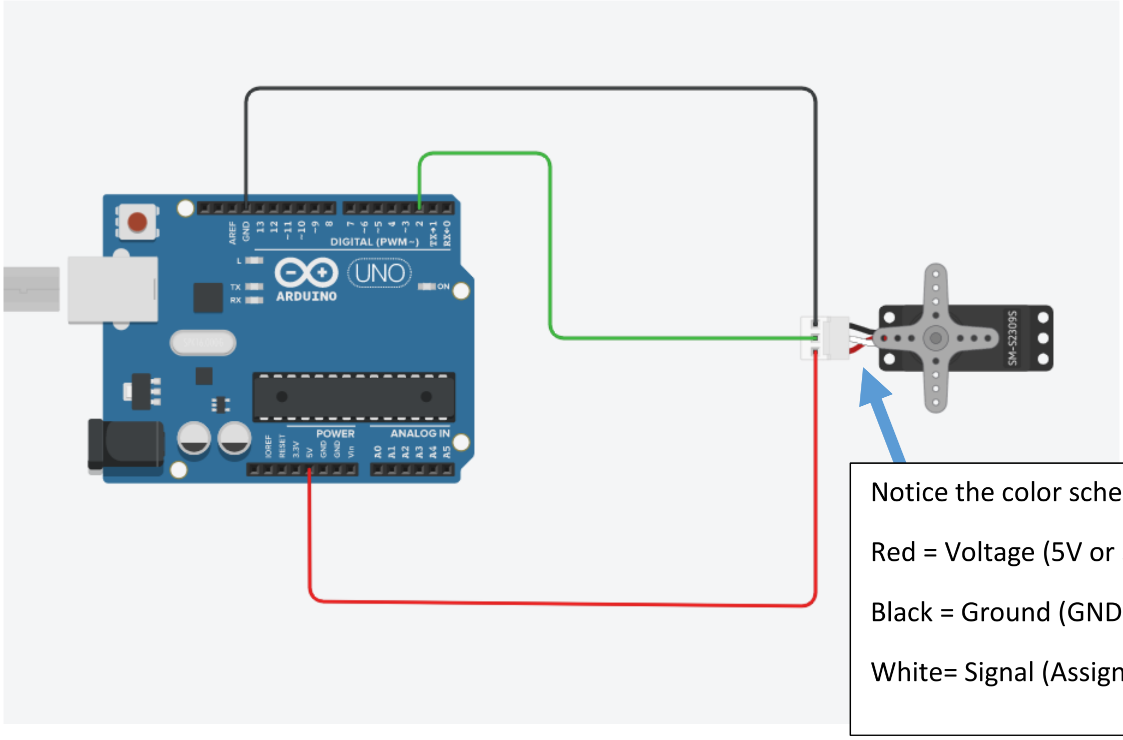


Micro Servo

Name 1

Type Continuous

Basic Wiring Setup



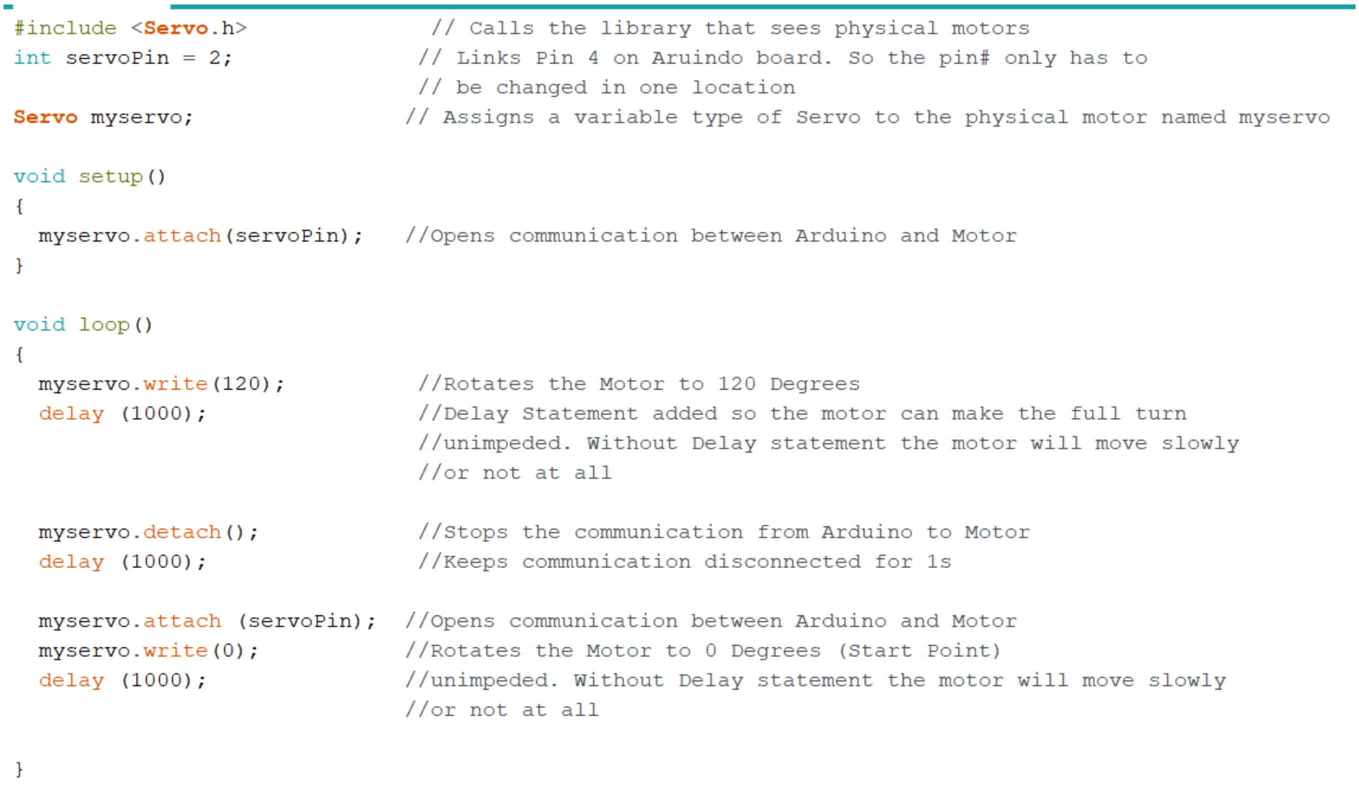
Notice the color scheme

Red = Voltage (5V or 3.3V)

Black = Ground (GND)

White= Signal (Assign to Digital Ports)

2. Write the following Program



```
#include <Servo.h>                  // Calls the library that sees physical motors
int servoPin = 2;                  // Links Pin 4 on Aruindo board. So the pin# only has to
                                   // be changed in one location
Servo myservo;                    // Assigns a variable type of Servo to the physical motor named myservo

void setup()
{
  myservo.attach(servoPin);   //Opens communication between Arduino and Motor
}

void loop()
{
  myservo.write(120);           //Rotates the Motor to 120 Degrees
  delay (1000);                 //Delay Statement added so the motor can make the full turn
                                //unimpeded. Without Delay statement the motor will move slowly
                                //or not at all

  myservo.detach();             //Stops the communication from Arduino to Motor
  delay (1000);                 //Keeps communication disconnected for 1s

  myservo.attach (servoPin);  //Opens communication between Arduino and Motor
  myservo.write(0);           //Rotates the Motor to 0 Degrees (Start Point)
  delay (1000);               //unimpeded. Without Delay statement the motor will move slowly
                              //or not at all

}
```

3. Run Simulation Notice the Bracket turning to 120 Degrees > paused > Rotates back to 0 Degrees
4. Stop Simulation >Notice the Bracket on the motor resets to 0 degrees
5. Modify the degrees

Submission: Email teacher URL under the Share Icon.

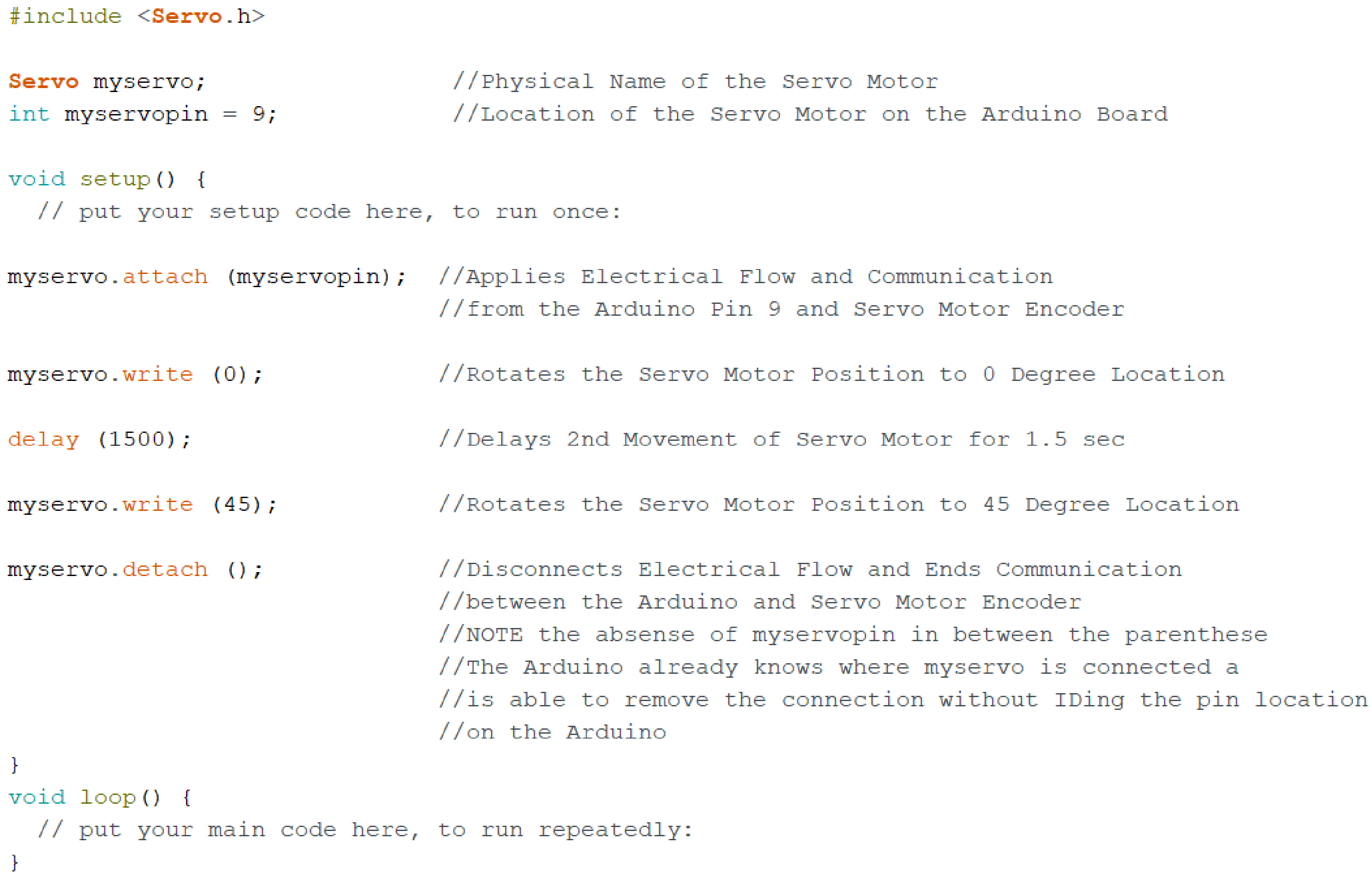
Program 1: 360 Degree Motor

Objective: Servo Motor Setting Specific Angle Position in Code.
Wire the Motor as Shown using a Sensor Shield

Motor Type: Servo Motor set to Positional

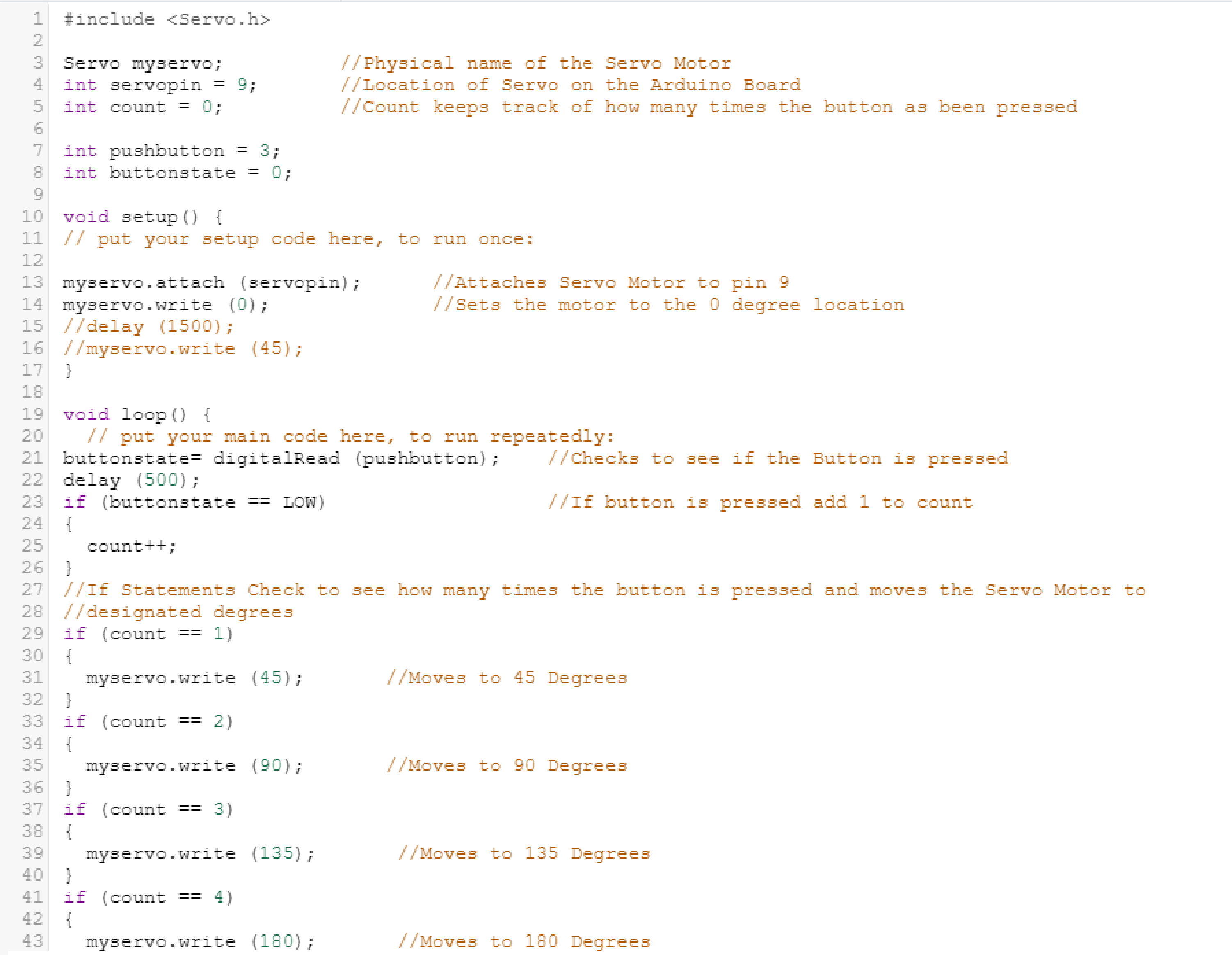
Arduino Motor Location: Digital Pin 9

Code: Moving 360 Degree Motor
Step 1: Code will set the motor to 0 degrees wait 1.5 seconds move to 45 Degrees



```
#include <Servo.h>

Servo myservo;                 //Physical Name of the Servo Motor
int myservopin = 9;            //Location of the Servo Motor on the Arduino Board

void setup() {
  // put your setup code here, to run once:

myservo.attach (myservopin);  //Applies Electrical Flow and Communication
                              //from the Arduino Pin 9 and Servo Motor Encoder

myservo.write (0);            //Rotates the Servo Motor Position to 0 Degree Location

delay (1500);                 //Delays 2nd Movement of Servo Motor for 1.5 sec

myservo.write (45);           //Rotates the Servo Motor Position to 45 Degree Location

myservo.detach ();            //Disconnects Electrical Flow and Ends Communication
                              //between the Arduino and Servo Motor Encoder
                              //NOTE the absense of myservopin in between the parenthese
                              //The Arduino already knows where myservo is connected a
                              //is able to remove the connection without IDing the pin location
                              //on the Arduino
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Step 2: Adding a Push Button
Note: Step 1 Code that is commented out
Component to Circuit: Add a push button and push button code as shown
NOTE: Code written with push button module, low = pressed and high = not-pressed



```
#include <Servo.h>

Servo myservo;          //Physical name of the Servo Motor
int servopin = 9;       //Location of Servo on the Arduino Board
int count = 0;          //Count keeps track of how many times the button as been pressed

int pushbutton = 3;
int buttonstate = 0;

void setup() {
// put your setup code here, to run once:

myservo.attach (servopin);      //Attaches Servo Motor to pin 9
myservo.write (0);              //Sets the motor to the 0 degree location
//delay (1500);
//myservo.write (45);
}

void loop() {
  // put your main code here, to run repeatedly:
buttonstate= digitalRead (pushbutton);    //Checks to see if the Button is pressed
delay (500);
if (buttonstate == LOW)                   //If button is pressed add 1 to count
{
  count++;
}
//If Statements Check to see how many times the button is pressed and moves the Servo Motor to
//designated degrees
if (count == 1)
{
  myservo.write (45);       //Moves to 45 Degrees
}
if (count == 2)
{
  myservo.write (90);       //Moves to 90 Degrees
}
if (count == 3)
{
  myservo.write (135);       //Moves to 135 Degrees
}
if (count == 4)
{
  myservo.write (180);       //Moves to 180 Degrees
```

Assignment:

1. Serial Monitor Output for **ALL** button presses
    a. Output Button Press
    b. Output Degree Location
2. 5th Press have the program do the following
    - Ask what degree user would like to go too
      (See Assignment 1: Volume for Input Code)
    - Servo Motor goes to user input
3. 6th Press
    - Reset count to run the program again
    - Output "Last Press, Time to Reset"

Submission: Email > To: Teachers Email > Fill Subject Heading TinkerCAD "Tutorial or Assignment" "Tutorial/Assignment Name" > Send

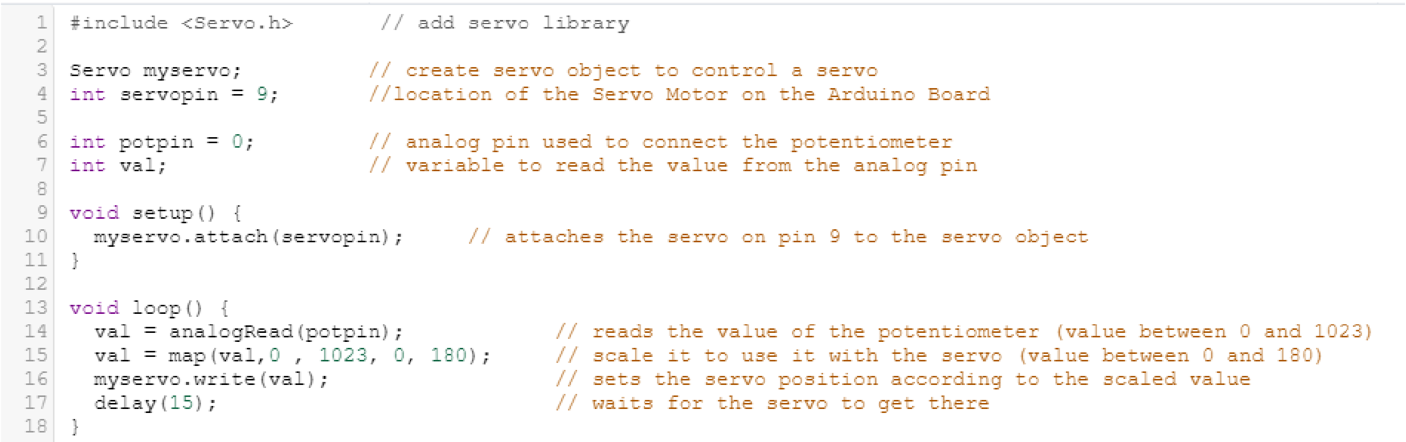
Objective: Control Servo Motor by using a Potentiometer.

Operation: Rotate the Potentiometer to change degree, the Servo Motor will use the degree angle of rotation from the Potentiometer to rotate the Servo Motor to the same degree.

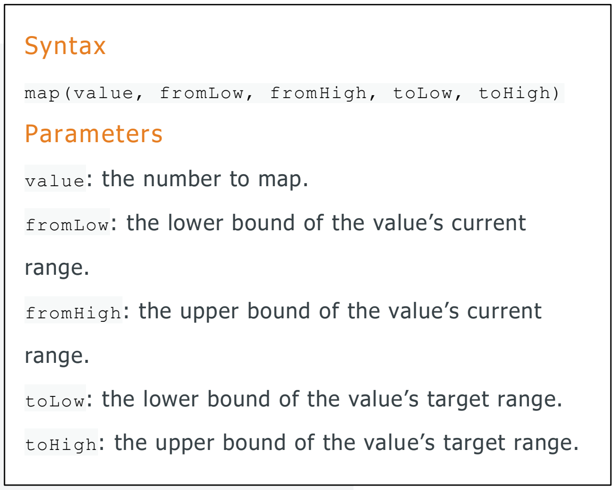
Motor Type: Servo Motor set to Positional

Arduino Motor Location: Digital Pin 9

Code:



```
#include <Servo.h>          // add servo library

Servo myservo;             // create servo object to control a servo
int servopin = 9;          //location of the Servo Motor on the Arduino Board

int potpin = 0;            // analog pin used to connect the potentiometer
int val;                   // variable to read the value from the analog pin

void setup() {
  myservo.attach(servopin);     // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val,0 , 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```



Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameters

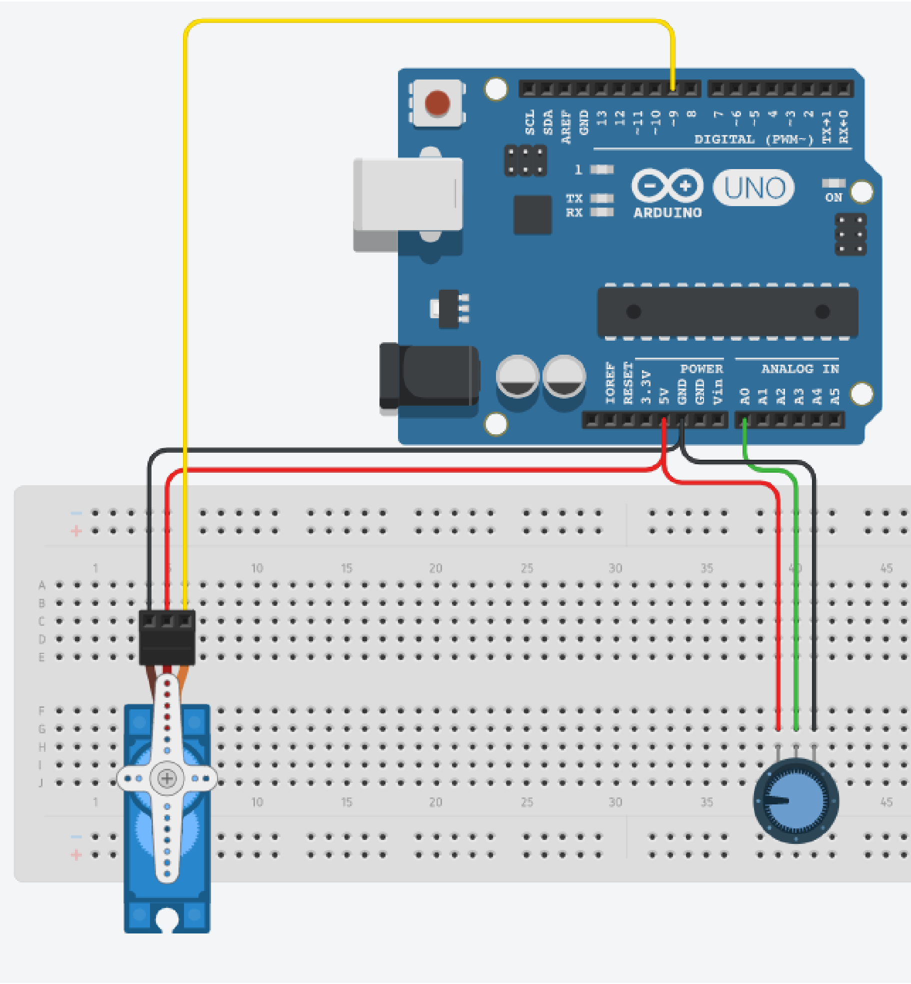
`value`: the number to map.

`fromLow`: the lower bound of the value's current range.

`fromHigh`: the upper bound of the value's current range.

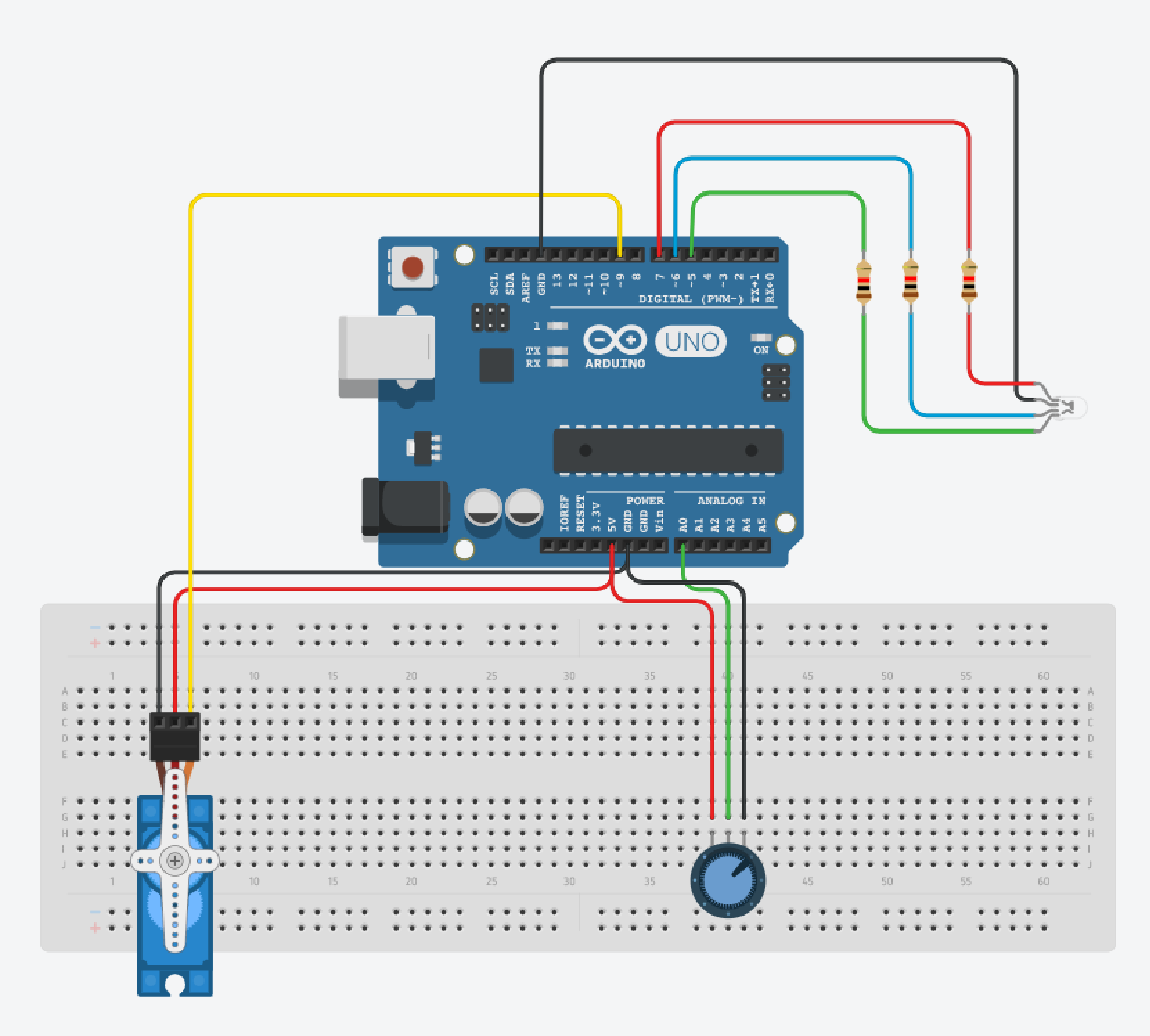
`toLow`: the lower bound of the value's target range.

`toHigh`: the upper bound of the value's target range.

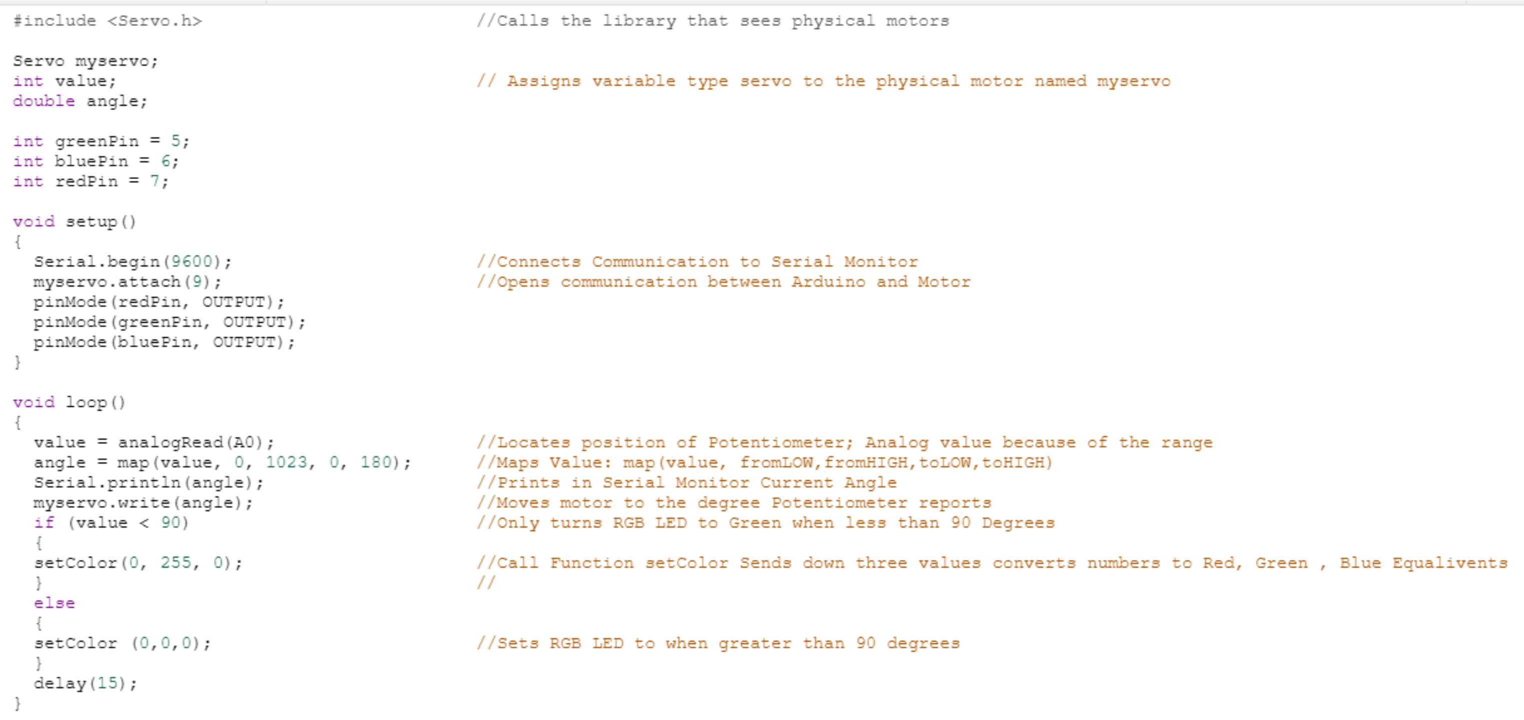


Adding RGB LED

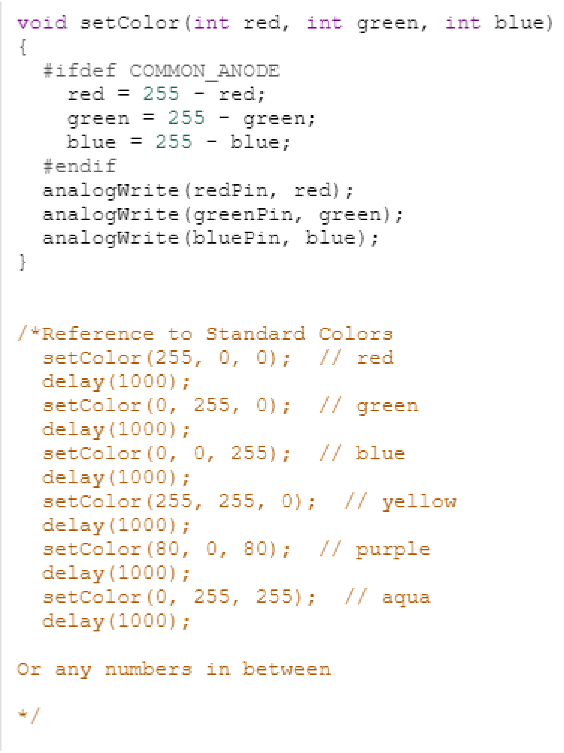
1. Add RGB LED and 3 1K Ohm Resistors as shown



2. Modify the Code > Adding to the code
    a. Variable Redpin, Bluepin, and Greenpin
    b. Calling Function setColor to assign color values to the RGB LED
    c. Creating Function setColor (see reference document describing



```
#include <Servo.h>                          //Calls the library that sees physical motors

Servo myservo;
int value;                                  // Assigns variable type servo to the physical motor named myservo
double angle;

int greenPin = 5;
int bluePin = 6;
int redPin = 7;

void setup()
{
  Serial.begin(9600);                       //Connects Communication to Serial Monitor
  myservo.attach(9);                        //Opens communication between Arduino and Motor
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop()
{
  value = analogRead(A0);                   //Locates position of Potentiometer; Analog value because of the range
  angle = map(value, 0, 1023, 0, 180);      //Maps Value: map(value, fromLOW,fromHIGH,toLOW,toHIGH)
  Serial.println(angle);                    //Prints in Serial Monitor Current Angle
  myservo.write(angle);                     //Moves motor to the degree Potentiometer reports
  if (value < 90)                           //Only turns RGB LED to Green when less than 90 Degrees
  {
  setColor(0, 255, 0);                      //Call Function setColor Sends down three values converts numbers to Red, Green , Blue Equalivents
  }                                         //
  else
  {
  setColor (0,0,0);                         //Sets RGB LED to when greater than 90 degrees
  }
  delay(15);
}
```



```
void setColor(int red, int green, int blue)
{
  #ifdef COMMON_ANODE
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
  #endif
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}


/*Reference to Standard Colors
  setColor(255, 0, 0);  // red
  delay(1000);
  setColor(0, 255, 0);  // green
  delay(1000);
  setColor(0, 0, 255);  // blue
  delay(1000);
  setColor(255, 255, 0);  // yellow
  delay(1000);
  setColor(80, 0, 80);  // purple
  delay(1000);
  setColor(0, 255, 255);  // aqua
  delay(1000);

Or any numbers in between

*/
```

3. Run the simulation > Move the Potentiometer by placing the cursor on the know and rotating

Assignment

1. Add a Slide Switch to do the following
    a. On State:
        i. Motor will move based on the turning of the potentiometer
        ii. Colors based on degree
            1. Gray: 0
            2. Red: 0 < value <= 60
            3. Blue: 60 < value <= 100
            4. Yellow: 100 < value <= 130
            5. Green: 130 < value < 180
            6. Clear: 180

            Hint: Ranges in if statement if (value > 0 && value <= 60) then ….. Remember the programs can only do a 1 for 1 comparison.

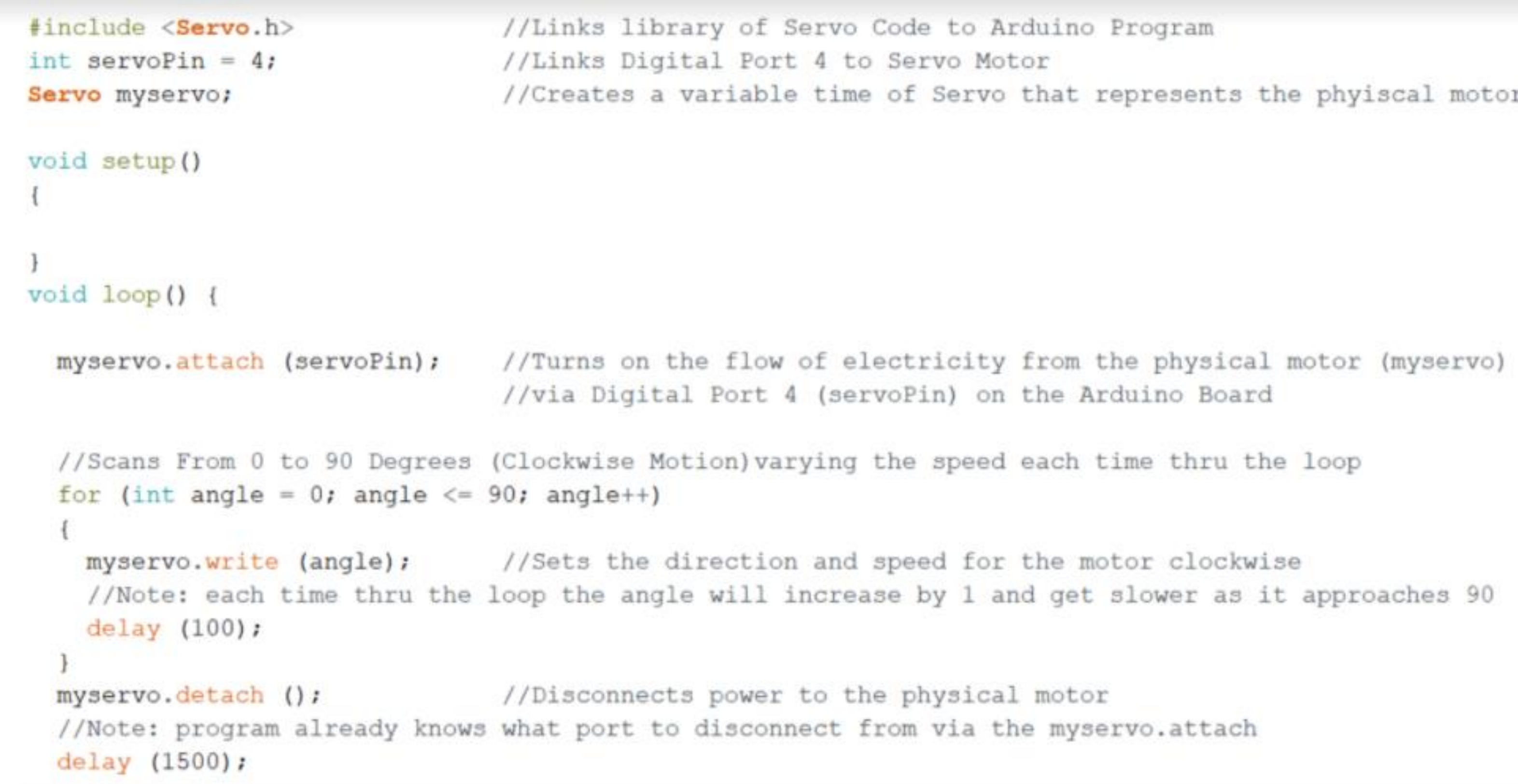
    b. Off State:
            1. Motor Off
            2. Color: Clear

Submission: To Submit TinkerCAD Tutorials and Assignments: Select Share Icon (Top Right Corner) > Select Invite People > Copy URL > Navigate to Student Email Account > Compose a New Email > To: Teachers Email > Fill Subject Heading TinkerCAD "Tutorial or Assignment" "Tutorial/Assignment Name" > Send

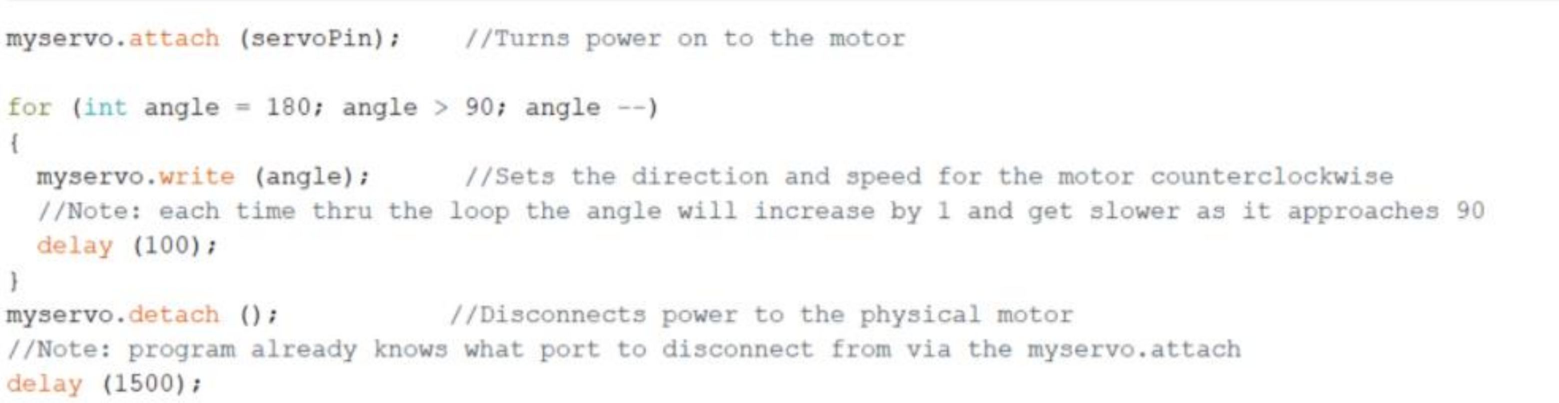
Program 3: 360 Degree Continuous Servo
Objective: Servo Motor Sweeping Angles Clockwise versus Counter-wise Spin Wire the Motor

Motor Type: Servo Motor set to Continuous
Arduino Motor Location: Digital Pin 4

Code:



```
#include <Servo.h>                //Links library of Servo Code to Arduino Program
int servoPin = 4;                 //Links Digital Port 4 to Servo Motor
Servo myservo;                    //Creates a variable time of Servo that represents the phyiscal motor

void setup()
{

}
void loop() {

  myservo.attach (servoPin);      //Turns on the flow of electricity from the physical motor (myservo)
                                  //via Digital Port 4 (servoPin) on the Arduino Board

  //Scans From 0 to 90 Degrees (Clockwise Motion)varying the speed each time thru the loop
  for (int angle = 0; angle <= 90; angle++)
  {
    myservo.write (angle);        //Sets the direction and speed for the motor clockwise
    //Note: each time thru the loop the angle will increase by 1 and get slower as it approaches 90
    delay (100);
  }
  myservo.detach ();              //Disconnects power to the physical motor
  //Note: program already knows what port to disconnect from via the myservo.attach
  delay (1500);
```



```
myservo.attach (servoPin);     //Turns power on to the motor

for (int angle = 180; angle > 90; angle --)
{
  myservo.write (angle);       //Sets the direction and speed for the motor counterclockwise
  //Note: each time thru the loop the angle will increase by 1 and get slower as it approaches 90
  delay (100);
}
myservo.detach ();            //Disconnects power to the physical motor
//Note: program already knows what port to disconnect from via the myservo.attach
delay (1500);
```

Submission: To Submit TinkerCAD Tutorials and Assignments: Select Share Icon (Top Right Corner) > Select Invite People > Copy URL > Navigate to Student Email Account > Compose a New Email > To: Teachers Email > Fill Subject Heading TinkerCAD "Tutorial or Assignment" "Tutorial/Assignment Name" > Send

Program 4: 360 Degree Continuous Servo Motor with Ultrasonic Sensor

Objective: Control the On/Off of the 360 Degree Continuous Servo Motor based on a sensor value as opposed to a user assigned value.

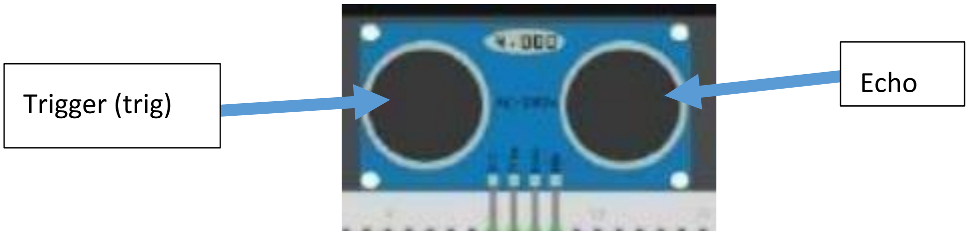
Motor Type: Servo Motor set to Continuous

Arduino Motor Location: Digital Pin 2

Ultrasonic Sensor: See Below

Ultrasonic Sensor is designed to send out a sound wave signal called the Trigger; and receive the bounced back sound wave into the Echo port. The sound wave will pulsate the Trigger on and off so the sound wave returning from the contacted object will be able to pass between the pulses. If the Trigger was constantly on the returning sound wave would be distorted.

Trigger Sound Wave will be a conical shape and can be distorted from ambient noise and materials that absorb sound (i.e cardboard, tennis ball, etc.)



Sound Calculation Formula: Distance L = 1/2 × T × C

Where L is the distance, T is the time between the emission and reception, and C is the sonic speed. (The value is multiplied by 1/2 because T is the time for go-and-return distance.)

Ultrasonic Sensor- Attaching to Arduino

Wires: Use Four (4) Female to Female jumper wire connectors

Wire the Following

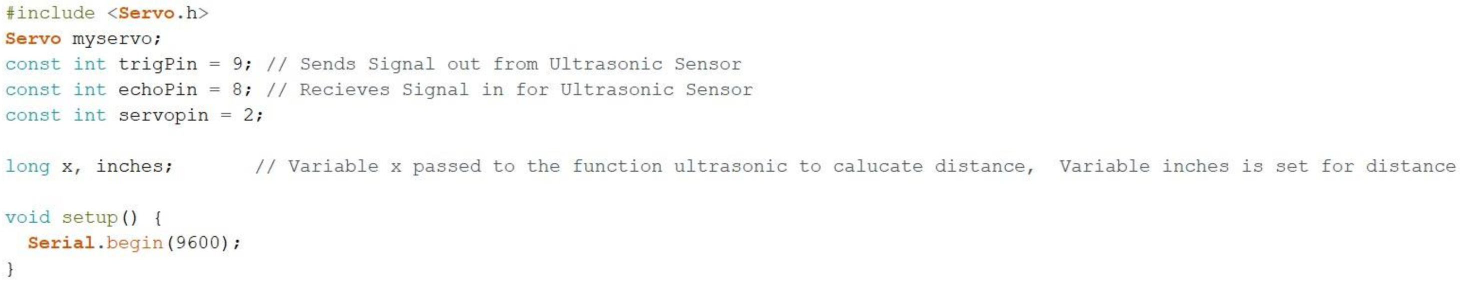
Follow the proper color set

Ultrasonic Sensor	Arduino Board	Pin Row
Ground (GND)	G	9
Volt (VCC)	V	9
Echo	S	8
Trig	S	9

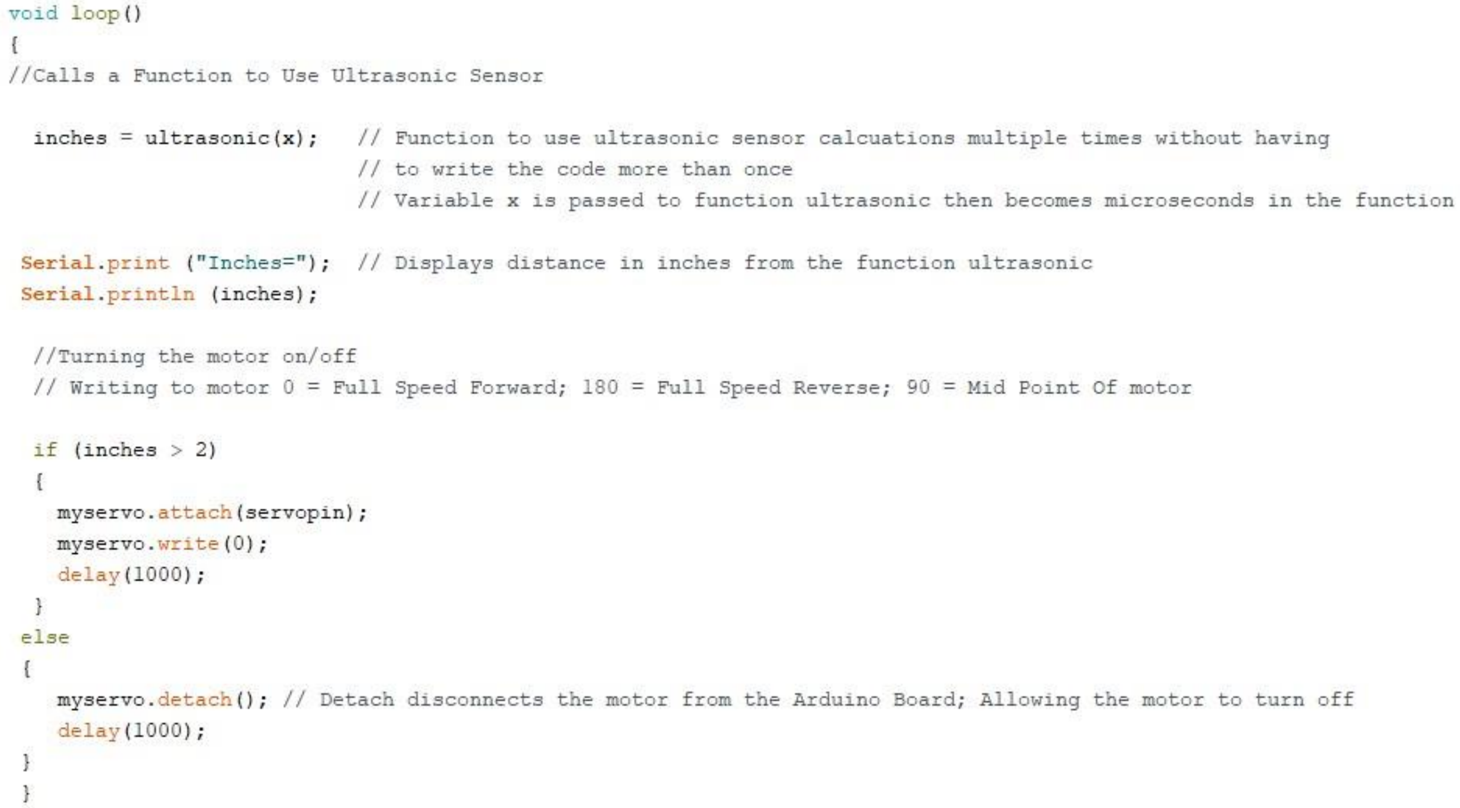
NOTE: Voltage and Ground Wires must be in the same digital port row in order to complete the circuit with the Arduino board and the Ultrasonic Sensor circuit board. The Trig and Echo Pins can be located anywhere in the Signal Column (I.E Echo in port 9 and Trig in port 10).

Write the Following Code

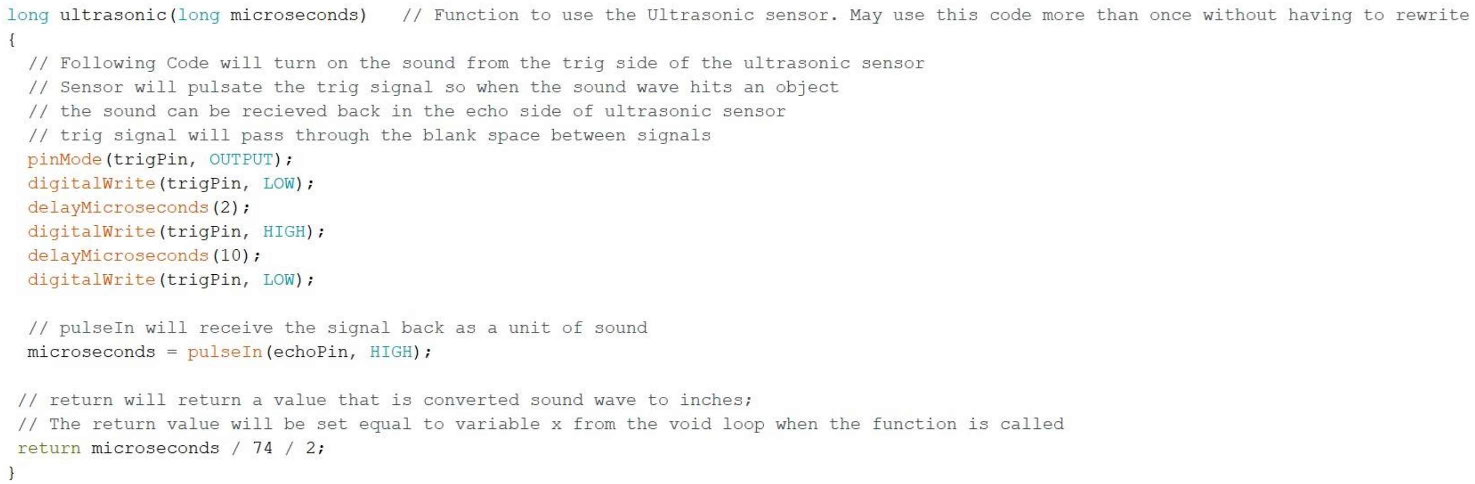
NOTE: Ultrasonic Code is using a function to consolidate code in case the user needs to read the ultrasonic distance in multiple locations of the program (See Tutorial/Assignment 4: Ultrasonic for Function details)



```
#include <Servo.h>
Servo myservo;
const int trigPin = 9; // Sends Signal out from Ultrasonic Sensor
const int echoPin = 8; // Recieves Signal in for Ultrasonic Sensor
const int servopin = 2;

long x, inches;       // Variable x passed to the function ultrasonic to calucate distance,  Variable inches is set for distance

void setup() {
  Serial.begin(9600);
}
```



```
void loop()
{
//Calls a Function to Use Ultrasonic Sensor

  inches = ultrasonic(x);   // Function to use ultrasonic sensor calcuations multiple times without having
                            // to write the code more than once
                            // Variable x is passed to function ultrasonic then becomes microseconds in the function

 Serial.print ("Inches=");  // Displays distance in inches from the function ultrasonic
 Serial.println (inches);

  //Turning the motor on/off
  // Writing to motor 0 = Full Speed Forward; 180 = Full Speed Reverse; 90 = Mid Point Of motor

  if (inches > 2)
  {
    myservo.attach(servopin);
    myservo.write(0);
    delay(1000);
  }
 else
 {
    myservo.detach(); // Detach disconnects the motor from the Arduino Board; Allowing the motor to turn off
    delay(1000);
 }
 }
```



```
long ultrasonic(long microseconds)   // Function to use the Ultrasonic sensor. May use this code more than once without having to rewrite
{
  // Following Code will turn on the sound from the trig side of the ultrasonic sensor
  // Sensor will pulsate the trig signal so when the sound wave hits an object
  // the sound can be recieved back in the echo side of ultrasonic sensor
  // trig signal will pass through the blank space between signals
  pinMode(trigPin, OUTPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // pulseIn will receive the signal back as a unit of sound
  microseconds = pulseIn(echoPin, HIGH);

 // return will return a value that is converted sound wave to inches;
 // The return value will be set equal to variable x from the void loop when the function is called
 return microseconds / 74 / 2;
}
```

Assignment:

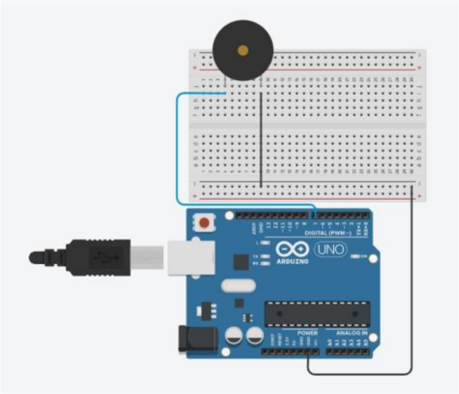
Add the following

1. Add LED RGB Light to turn

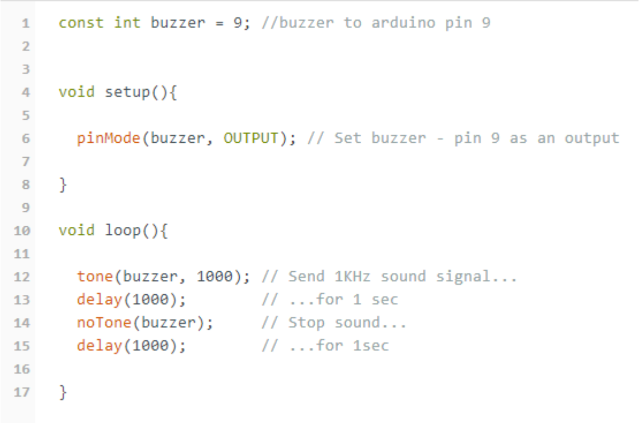
   Green: When distance is greater than 4 inches
   Yellow: When Distance is between 2 and 4 inches
   Red: When distance is less than 2 inches

   See Part 2 for RGB Code Function and Function Call

2. Piezo Buzzer
   a. **Definition:** initial mechanical motion is created by applying a voltage to a piezoelectric material, and this motion is converted into audible sound using diaphragms and resonators. User applies a hertz value to create the tone of the buzzer. User may also apply notes and note lengths similar to a speaker to provide a rhythm to the buzzer.
   b. Buzzer
      i. On: when ultrasonic sensor senses object less than 2 inches
      ii. OFF: when ultrasonic sensor senses object greater than 2 inches
   c. **Wiring**
   NOTE: Digital Port maybe located where user desires



   d. Code



```
const int buzzer = 9; //buzzer to arduino pin 9


void setup(){

  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(1000);        // ...for 1sec

}
```

Optional: Rickroll Piezo Program
Link: [Arduino.cc Project]()
(Hold Control Key > Then Left Click on Link)
Code: Uses notes and note lengths to provide the rhythm of the song. Be sure to copy all of the code.
NOTE: Code Lines 191- 193 makes the song repeat. This would be commented out.

Submission: To Submit TinkerCAD Tutorials and Assignments: Select Share Icon (Top Right Corner) > Select Invite People > Copy URL > Navigate to Student Email Account > Compose a New Email > To: [Teachers Email]() > Fill Subject Heading TinkerCAD "Tutorial or Assignment" "Tutorial/Assignment Name" > Send