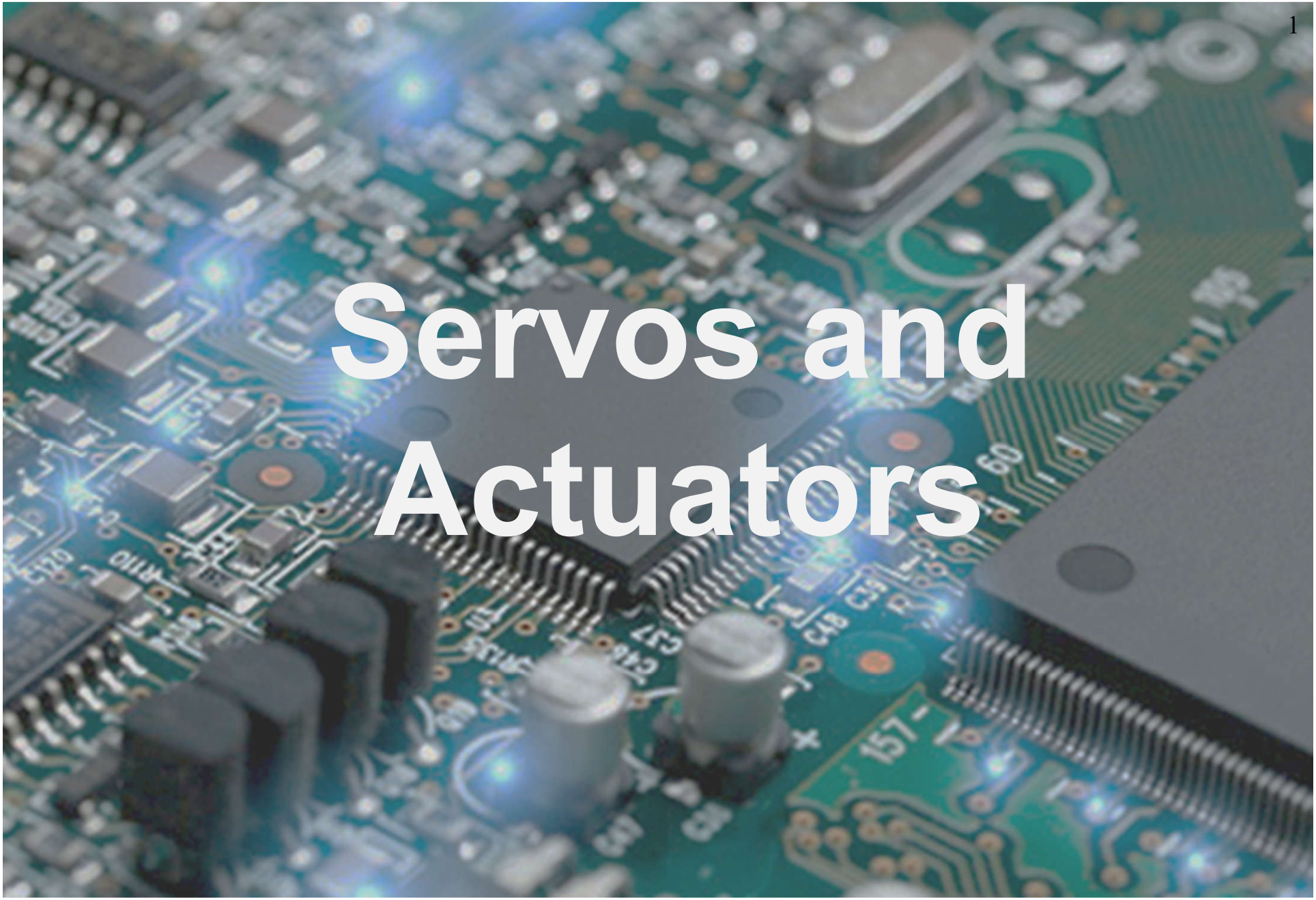


# Servos and Actuators

# Servos and Actuators

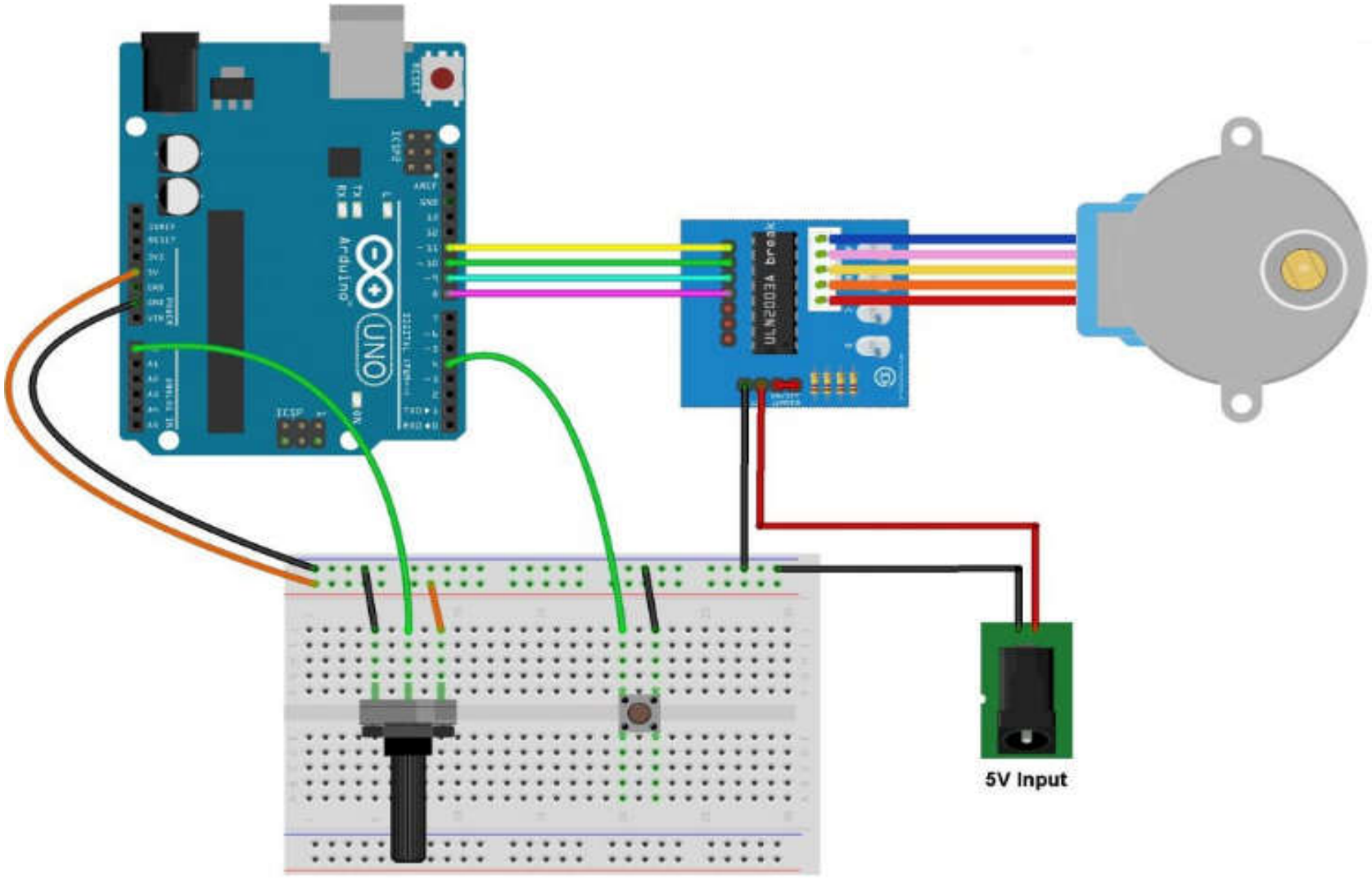
**Contents**

- Objectives
- Stepper Motor
- Servo Motor
- Robot Arm
- Piezo Alarm

# Objectives

- In this tutorial, you will learn:
  - To understand how to implement Stepper Motor.
  - To know how to make Servo motor project.
  - To become familiar using Robot arm project.
  - To be able to start Piezo.

# Stepper Motor Demo



# Stepper Motor Demo

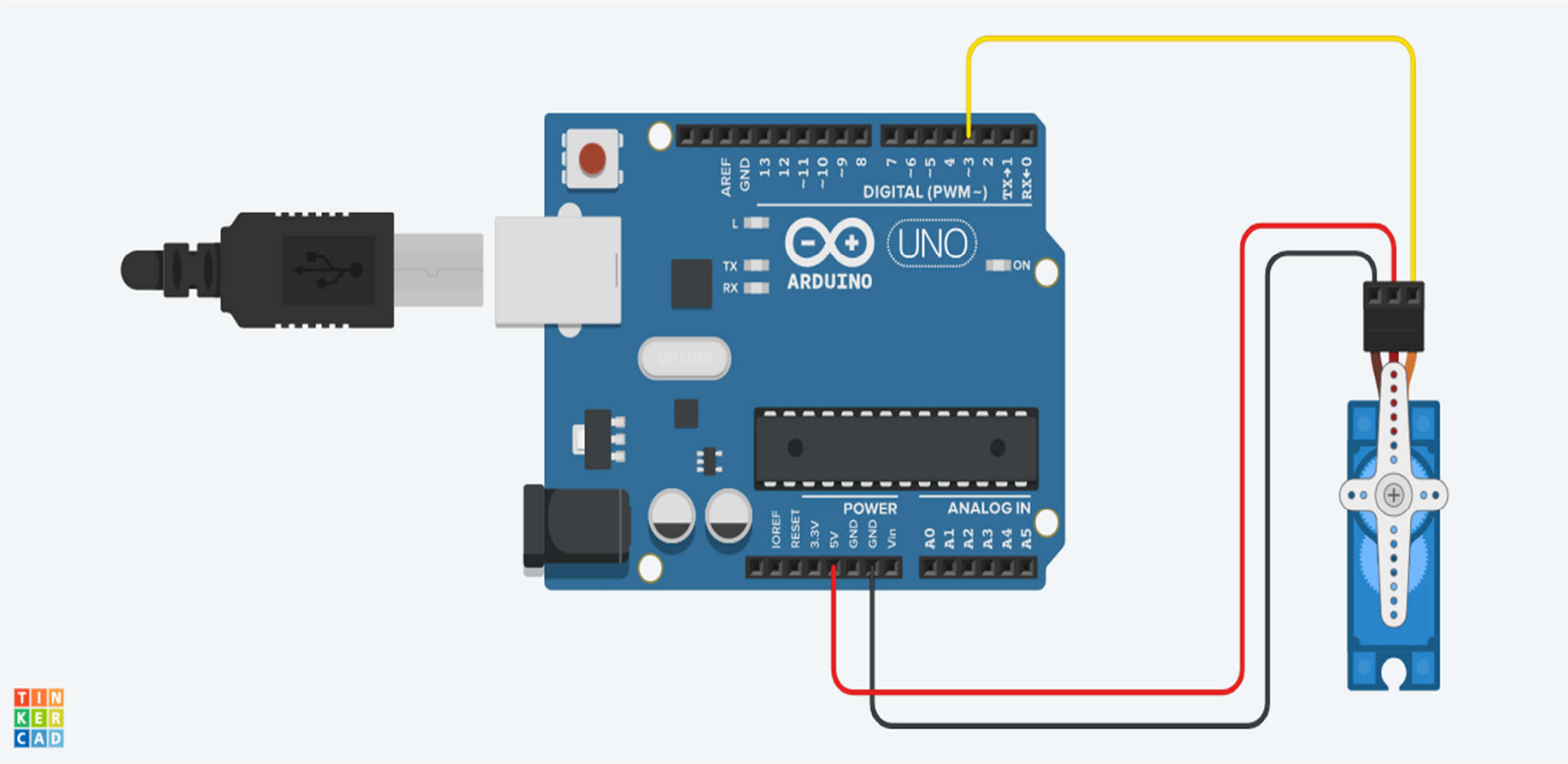
```
// include Arduino stepper motor library
#include <Stepper.h>

// change this to the number of steps on your motor
#define STEPS 32

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper(STEPS, 8, 10, 9, 11);

const int button =  4; // direction control button is connected to Arduino pin 4
const int pot    = A0; // speed control potentiometer is connected to analog pin 0
void setup()
{
  // configure button pin as input with internal pull up enabled
  pinMode(button, INPUT);
}

int direction_ = 1, speed_ = 0;

void loop()
{
  if ( digitalRead(button) == 0 )  // if button is pressed
    if ( debounce() )  // debounce button signal
    {
      direction_ *= -1;  // reverse direction variable
      while ( debounce() ) ;  // wait for button release
    }
```



# Stepper Motor Demo

```
// read analog value from the potentiometer
  int val = analogRead(pot);

  // map digital value from [0, 1023] to [2, 500]
  // ===> min speed = 2 and max speed = 500 rpm
  if ( speed_ != map(val, 0, 1023, 2, 100) )
  { // if the speed was changed
    speed_ = map(val, 0, 1023, 2, 100);
    // set the speed of the motor
    stepper.setSpeed(speed_);
  }

  // move the stepper motor
  stepper.step(direction_);

}

// a small function for button debounce
bool debounce()
{
  byte count = 0;
  for(byte i = 0; i < 5; i++) {
    if (digitalRead(button) == 0)
      count++;
    delay(10);
  }
  if(count > 2)  return 1;
  else           return 0;
}
```

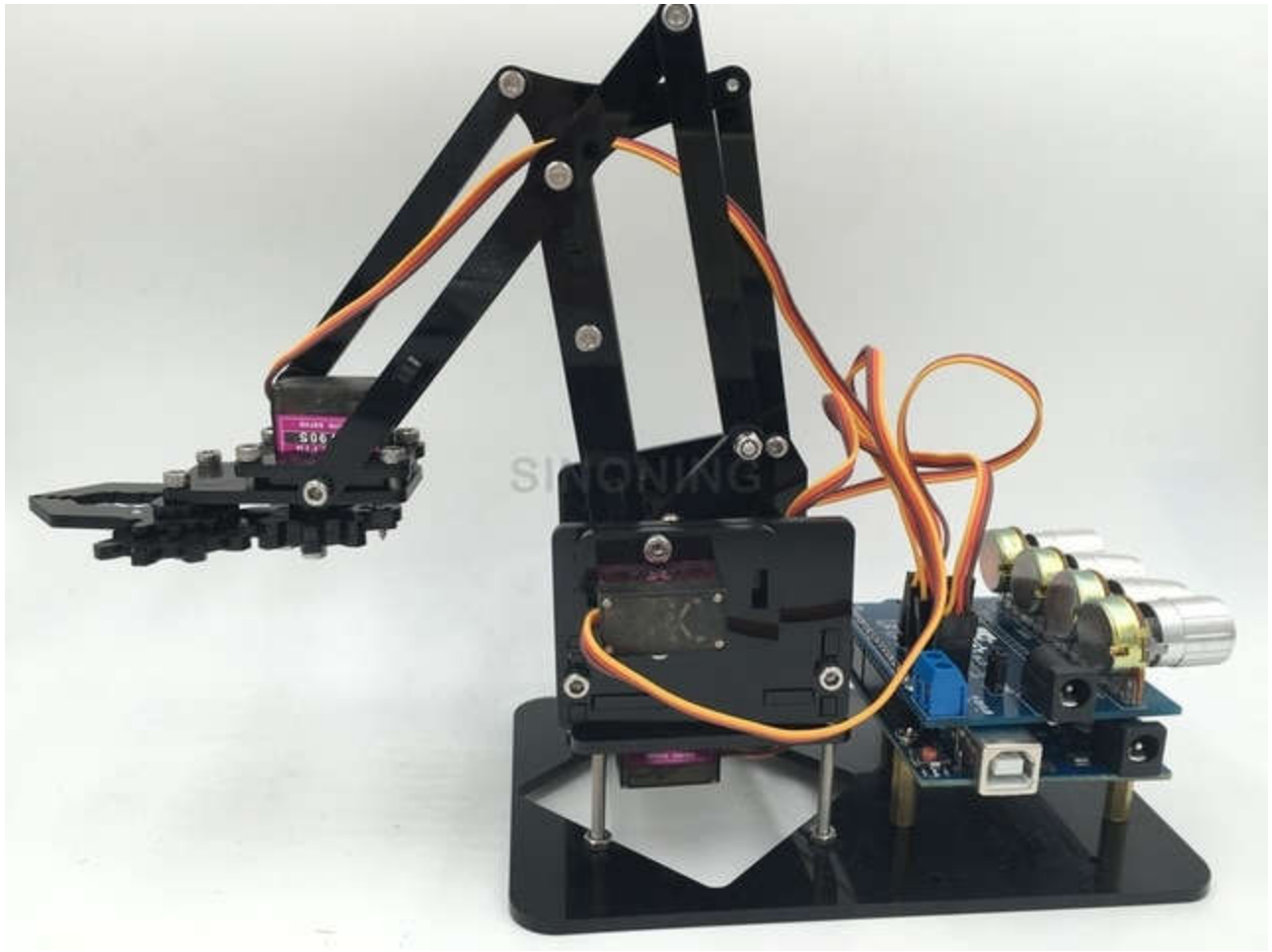
# Servo Motor Demo



# Servo Motor Demo

```
#include <Servo.h>
   Servo myservo; // create servo object to control a servo
   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(9); // 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);
}
```

# Robot Arm Demo



# Robot Arm Demo

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

Servo servo_test1;        //initialize a servo object for the connected servo
Servo servo_test2;
Servo servo_test3;
Servo servo_test4;

int angle = 0;

void setup()
{
  servo_test1.attach(11);      // attach the signal pin of servo to pin9 of arduino
  servo_test2.attach(10);
  servo_test3.attach(9);
  servo_test4.attach(6);
}
void loop()
{
  grip();
  up();
  waist1();
  down();
  grip();
  waist2();

}
```

# Robot Arm Demo

```
void grip()
{
  for(angle = 0; angle < 180; angle += 1)    // command to move from 0 degrees to 180 degrees
  {
    servo_test1.write(angle);                 //command to rotate the servo to the specified angle
    delay(15);
  }

  delay(1000);

  for(angle = 180; angle>=1; angle-=1)     // command to move from 180 degrees to 0 degrees
  {
    servo_test1.write(angle);              //command to rotate the servo to the specified angle
    delay(15);
  }

    delay(1000);

}
void waist1()
{
  for(angle = 0; angle < 180; angle += 1)    // command to move from 0 degrees to 180 degrees
  {
    servo_test3.write(angle);                 //command to rotate the servo to the specified angle
    delay(15);
  }

  delay(1000);

}
```

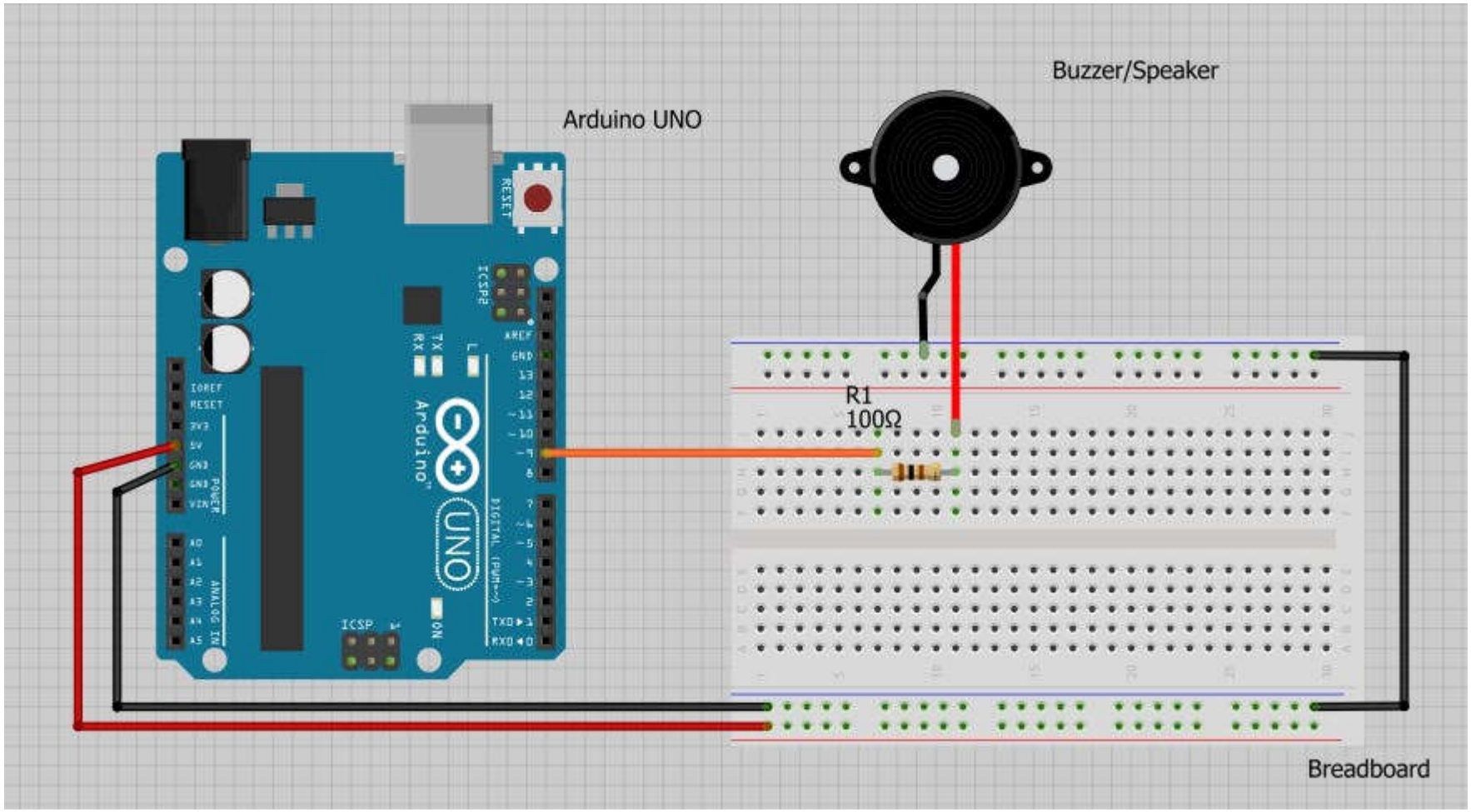
# Robot Arm Demo

```
void waist2()
{
for(angle = 180; angle>=1; angle-=1)     // command to move from 180 degrees to 0 degrees
  {
    servo_test3.write(angle);              //command to rotate the servo to the specified
angle
    delay(15);
  }

    delay(1000);

}
void up()
{
  for(angle = 0; angle < 180; angle += 1)    // command to move from 0 degrees to 180
degrees
  {
    servo_test2.write(angle);                //command to rotate the servo to the specified
angle
    delay(15);
  }

  delay(1000);

}
```

# Robot Arm Demo

```
void down()
{
for(angle = 180; angle>=1; angle-=1)     // command to move from 180 degrees to 0 degrees
  {
    servo_test2.write(angle);              //command to rotate the servo to the specified angle
    delay(15);
  }

    delay(1000);

}
```

# Piezo Demo



# Piezo Demo

```
#include "pitches.h"

// notes in the melody:
int melody[] = {

  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {

  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {

  // iterate over the notes of the melody:

  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.

    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

    int noteDuration = 1000 / noteDurations[thisNote];

    tone(9, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.

    // the note's duration + 30% seems to work well:

    int pauseBetweenNotes = noteDuration * 1.30;

    delay(pauseBetweenNotes);
```

# Piezo Demo

```
// stop the tone playing:

    noTone(9);

  }
}

void loop() {

  // no need to repeat the melody.
}
```

# THANK YOU !

# Q&A