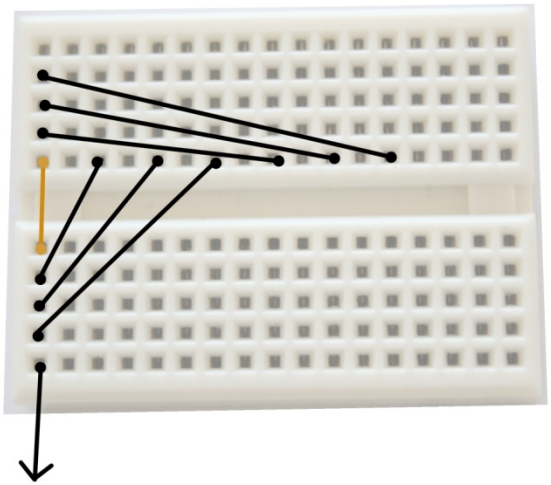
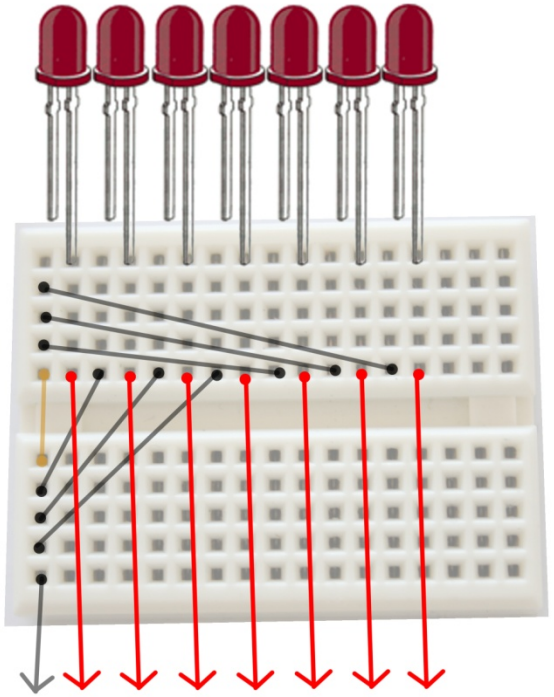
Extras

Arduino-Controlling Multiple Actuators separately from different output pins





Follow the above diagram to assemble your circuit:
Black represents the wiring that is connected to ground
Red represents wiring that is connected to Aurdoino output pins
Yellow represents wiring that is providing connections on Solderless Board to create seamless electricity flow for the Ground Line that we are creating. LEDs are seperately connected to digital pins while are all connected to the same Ground pin via a Ground Line on the solderless board

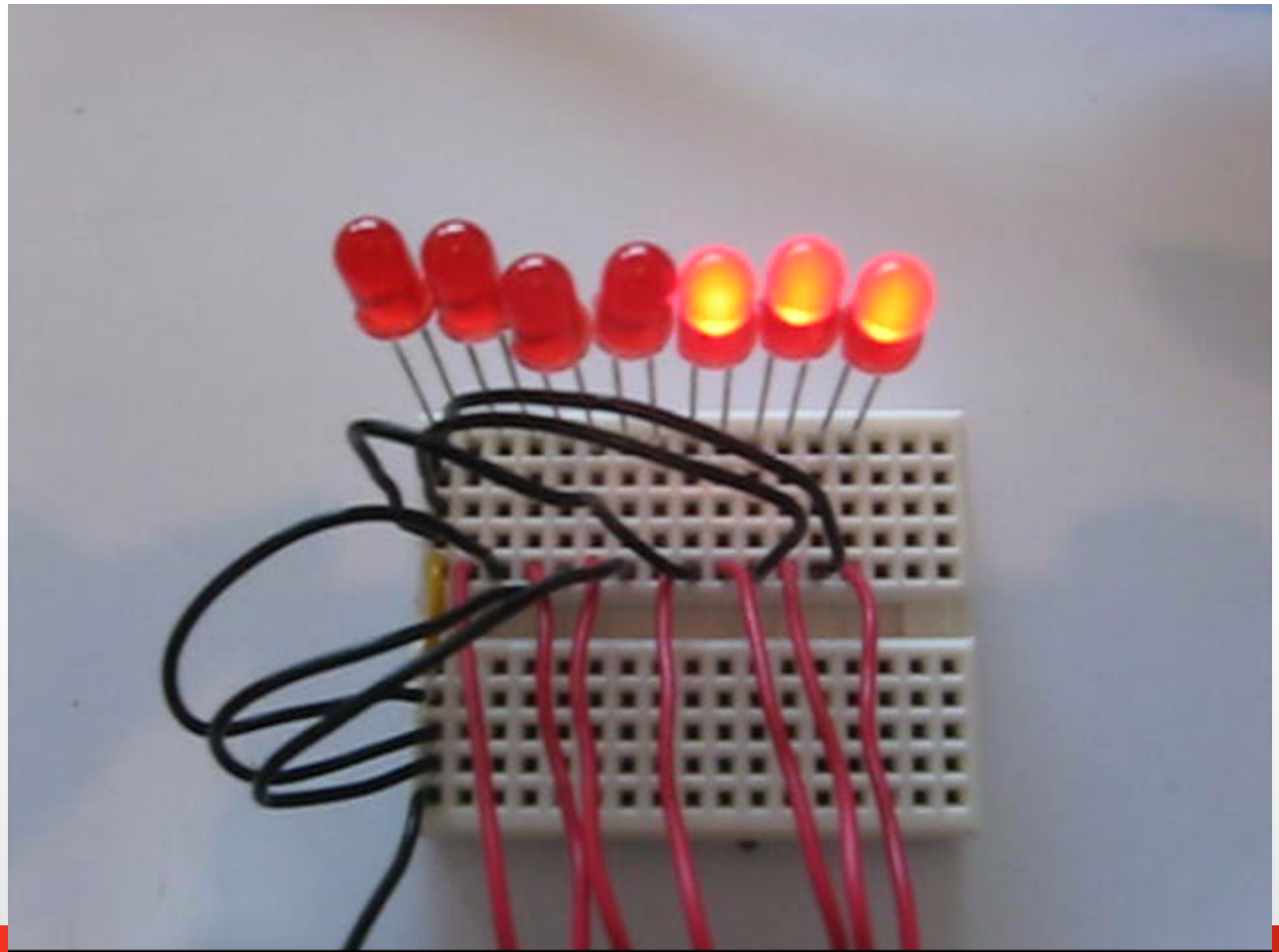
Arduino-Controlling Multiple Actuators separately from different output pins-Sequencing

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on one by one
  digitalWrite(i,HIGH); //Sending High Signal to Pin
  delay(1000); //Wait 1 second
  }
  for(int i=9; i>2; i--){//iterating through pin 8 to 2 and turning them off one by one
  digitalWrite(i,LOW); //Sending LOW Signal to Pin
  delay(1000); //Wait 1 second
  }
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Sequencing

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on one by one
  digitalWrite(i,HIGH); //Sending High Signal to Pin
  delay(1000); //Wait 1 second
  }
  for(int i=9; i>2; i--){//iterating through pin 8 to 2 and turning them off one by one
  digitalWrite(i,LOW); //Sending LOW Signal to Pin
  delay(1000); //Wait 1 second
  }
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Sequencing



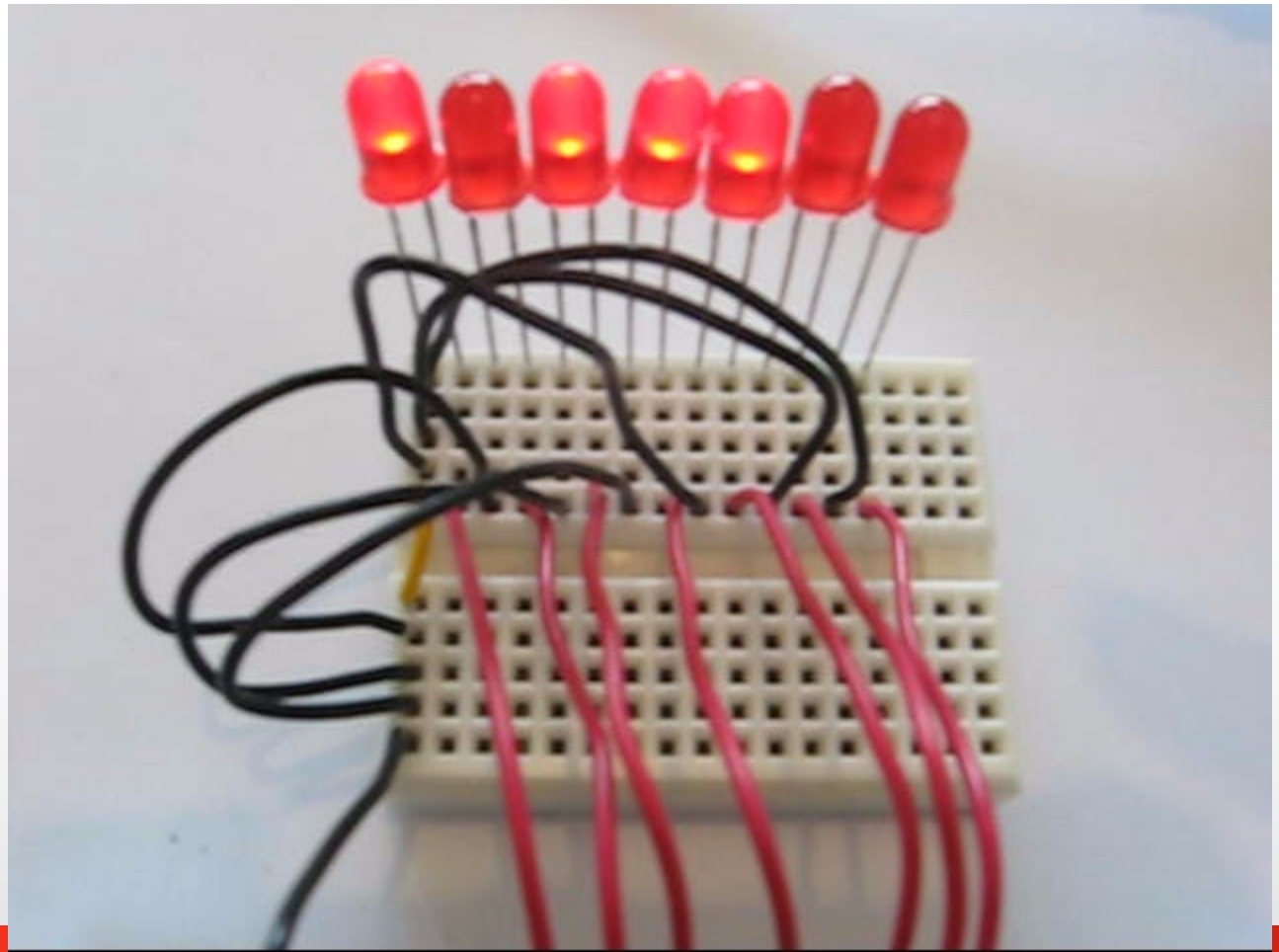
Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns



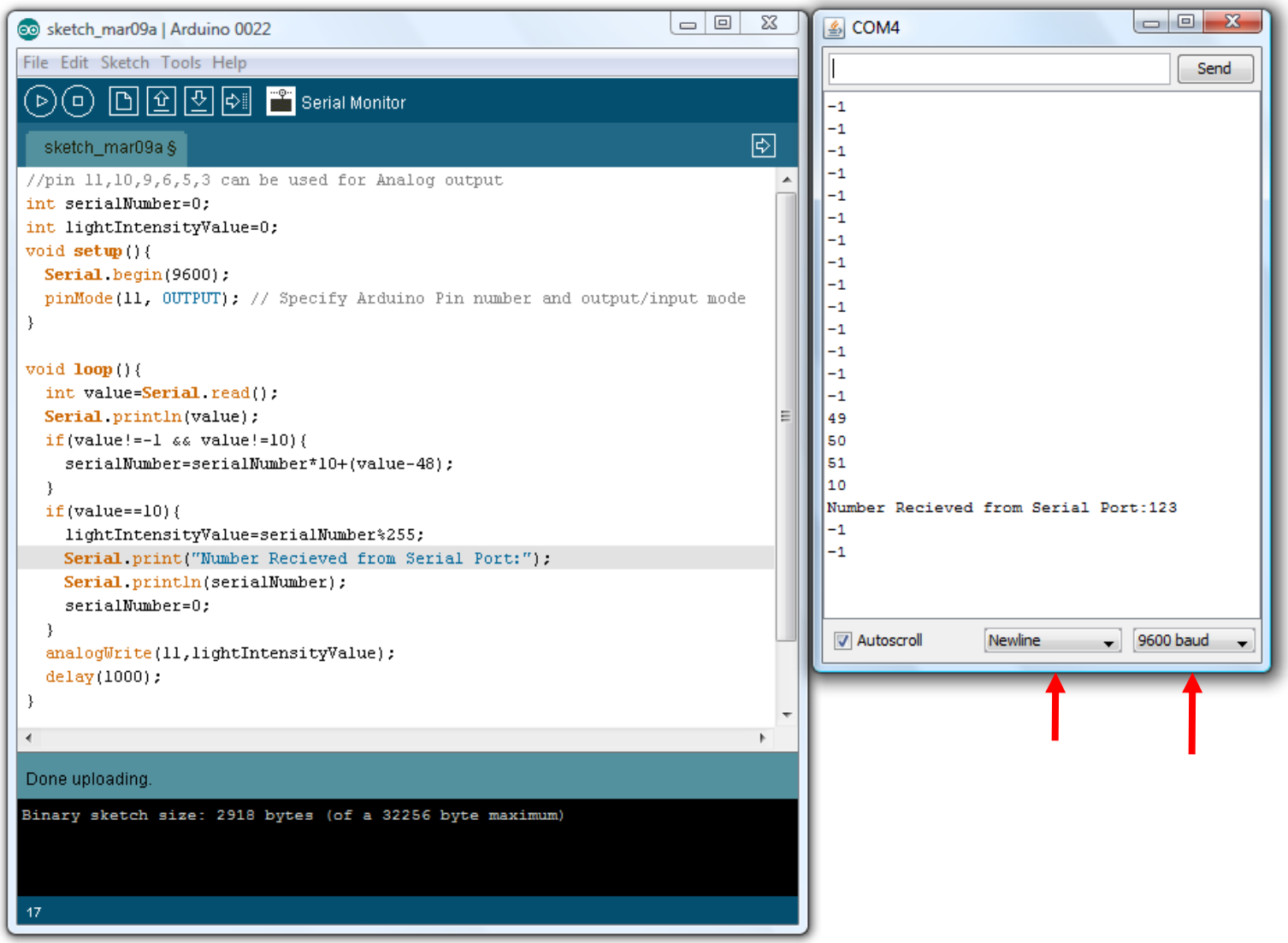
Arduino-Controlling Multiple Actuators separately from different output pins- Random Patterns

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on one by one
  digitalWrite(i,HIGH); //Sending High Signal to Pin
  delay(1000); //Wait 1 second
  }
  for(int i=9; i>2; i--){//iterating through pin 8 to 2 and turning them off one by one
  digitalWrite(i,LOW); //Sending LOW Signal to Pin
  delay(1000); //Wait 1 second
  }
}
```

Aside from introduction of randomness, pay attention to how changing the place of delay() function can change the systems behavior. Here we put the delay function out side of the for loop. As a result instead of seeing the change for each actuator one by one in a sequence, which is the case in the previous exercise, here, at first all the actuators(LEDs) are configured together and then the system pauses for one second to let us see the over all configuration.

Arduino-Controlling Actuators Based on Input from Arduino Serial Port



Arduino-Controlling Actuators Based on Input from Arduino Serial Port

```
//pin 11,10,9,6,5,3 can be used for Analog output
int serialNumber=0;
int lightIntensityValue=0;
void setup(){
  Serial.begin(9600);
  pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}

void loop(){
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    lightIntensityValue=serialNumber%255;
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
  }
  analogWrite(11,lightIntensityValue);
  delay(1000);
}
```

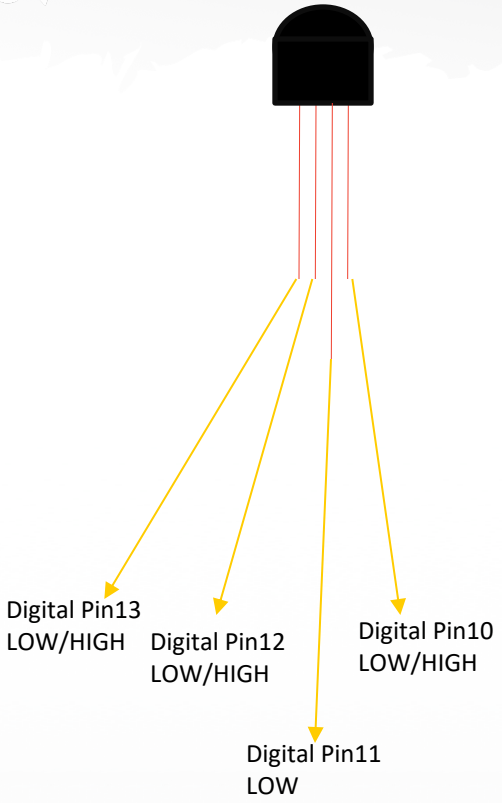
1. Data is received from Serial port as ASCII codes.
2. If data is numerical, each digit is sent separately.
3. ASCII code of zero is 48
4. To calculate the numerical value of a digit from its ASCI code: digit=**ASCII-48**
5. At the end of a package the serial port send a number **10**
6. If nothing is passed to the serial port, the port sends number **-1** as default

3-Color LED

```
void setup(){
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
}
void loop(){
digitalWrite(11,LOW);

digitalWrite(13,HIGH);
digitalWrite(12,LOW);
digitalWrite(10,LOW);
delay(1000);
digitalWrite(13,LOW);
digitalWrite(12,HIGH);
digitalWrite(10,LOW);
delay(1000);
digitalWrite(13,LOW);
digitalWrite(12,LOW);
digitalWrite(10,HIGH);
delay(1000);

}
```



**Make sure you are not doing the circuit vice versa!!!
** Sometimes the long leg should be high and the leg which is low would determine the color of the light

Arduino-
Digital Output-Sound-Piezo

A Piezo is an electronic piece that converts electricity energy to sound. It is a digital output device. You can make white noise or even exact musical notes (frequencies for musical notes) based on the duration that you iterate between HIGH and LOW signals.

A Piezo is a directional piece, meaning that it has a positive and negative pole. The positive pole should be connected to the digital output pin that you allocate to control the piezo and the negative pole should be connected to Ground pin

Arduino-Digital Output-Sound-Piezo

```
//connect piezo to pin 13 and ground
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  for(int i=0;i<8;i++){//iterating through notes
    for(int j=0;j<1000;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(freqs[i]);
    digitalWrite(13,LOW);
    delayMicroseconds(freqs[i]);
    }
  }
}
```

Arduino-Digital Output-Sound-Piezo

```
//connect piezo to pin 13 and ground
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  for(int i=0;i<8;i++){//iterating through notes
    for(int j=0;j<1000;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(freqs[i]);
    digitalWrite(13,LOW);
    delayMicroseconds(freqs[i]);
    }
  }
}
```

Arduino-Digital Output-Sound-Piezo-Playing a melody

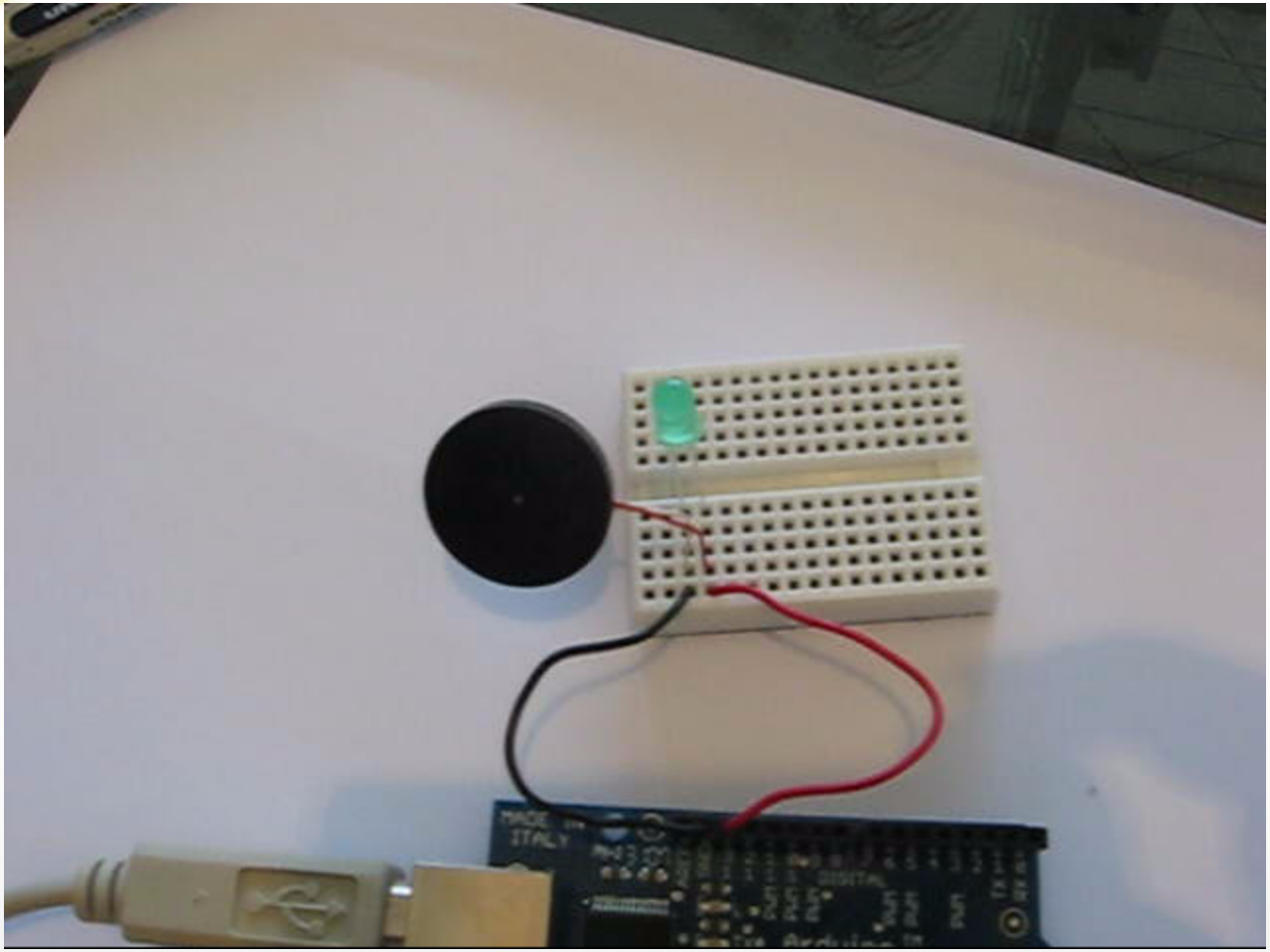
```
//connect piezo to pin 13 and ground
void playNote(int note)
{
  for(int j=0;j<60;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(note);
    digitalWrite(13,LOW);
    delayMicroseconds(note);
  }
  delay(60);
}
int pause=200;
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
//              i = { 0     1     2     3     4     5     6     7
//mi mi mi - mi mi mi - mi sol do re mi - - - fa fa fa fa fa mi mi mi mi re re mi re - sol - mi mi mi - mi mi mi - mi sol do re mi -- fa fa fa fa fa mi mi mi sol sol fa re do - - -
void setup(){
  pinMode(13,OUTPUT);


}
void loop(){
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);  delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);  delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);  playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);        delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);
  playNote(freqs[2]);  playNote(freqs[1]);  playNote(freqs[1]);  playNote(freqs[2]);
  playNote(freqs[1]);  delay(pause);        playNote(freqs[4]);  delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);  delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);  delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);  playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);        delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);
  playNote(freqs[4]);  playNote(freqs[4]);  playNote(freqs[3]);  playNote(freqs[3]);
  playNote(freqs[0]);  delay(pause);        delay(pause);        delay(pause);
}
```

Arduino-Digital Output-Sound-Piezo-Playing a melody

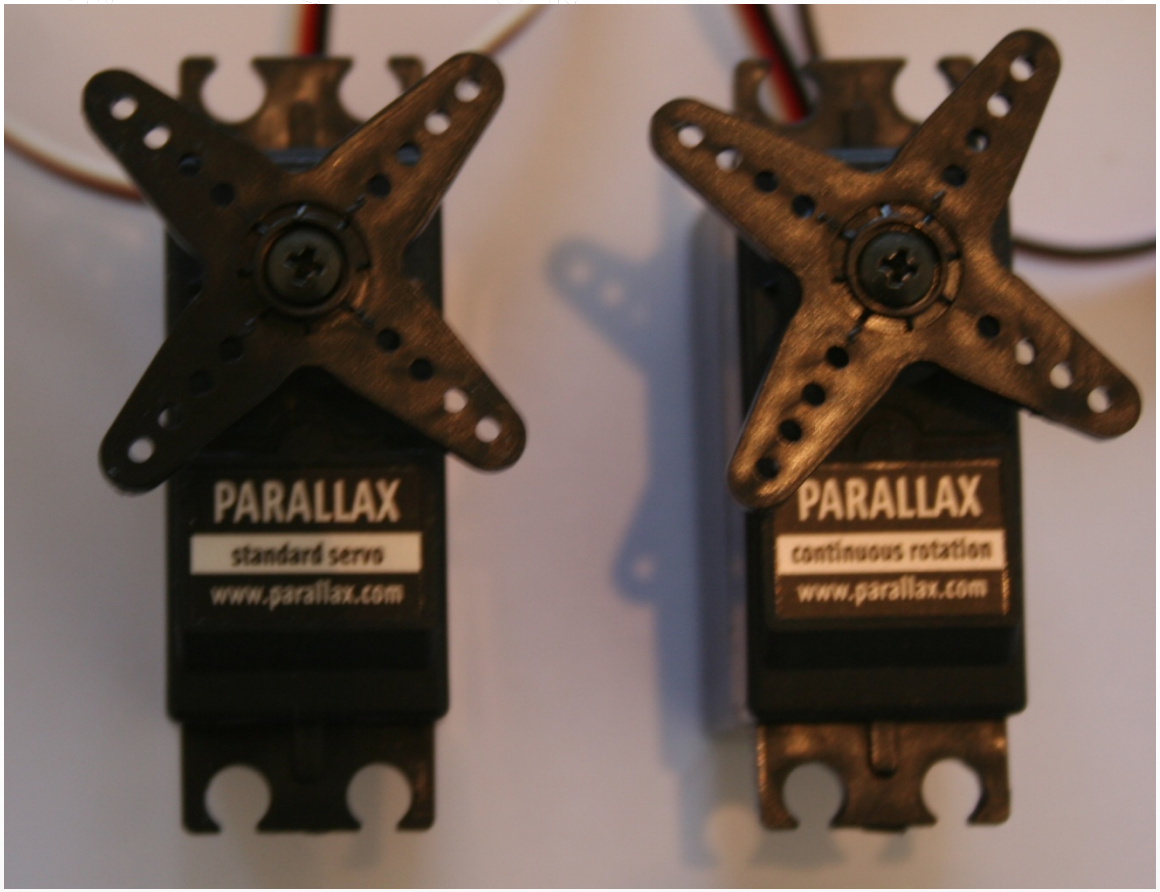
```
//connect piezo to pin 13 and ground
void playNote(int note)
{
  for(int j=0;j<60;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(note);
    digitalWrite(13,LOW);
    delayMicroseconds(note);
  }
  delay(60);
}
int pause=200;
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
//            i = { 0    1    2     3   4     5    6    7
//mi mi mi - mi mi mi - mi sol do re mi - - - fa fa fa fa fa mi mi mi mi re re mi re - sol - mi mi mi - mi mi mi - mi sol do re mi -- fa fa fa fa fa mi mi mi sol sol fa re do - - -
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);       delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[2]);  playNote(freqs[1]);  playNote(freqs[1]);   playNote(freqs[2]);
  playNote(freqs[1]);  delay(pause);       playNote(freqs[4]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);       delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[4]);  playNote(freqs[4]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[0]);  delay(pause);       delay(pause);        delay(pause);
}
```

Arduino- Same Signal Multiple Interpretations



In the same setting if you connect an LED parallel to Piezo, you can see how the same signal can be interpreted differently using a different output device that accept the same type of signals(in this case digital signal)

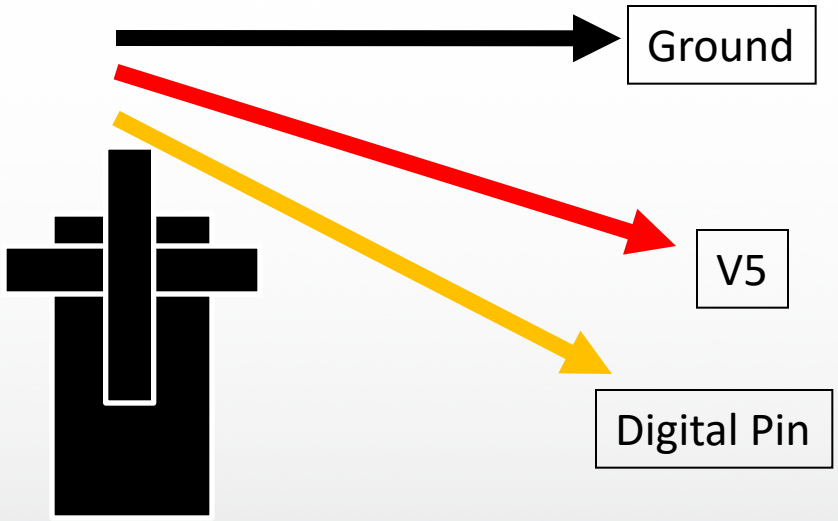
Arduino-DigitalOutput-Motion-Servo Motor

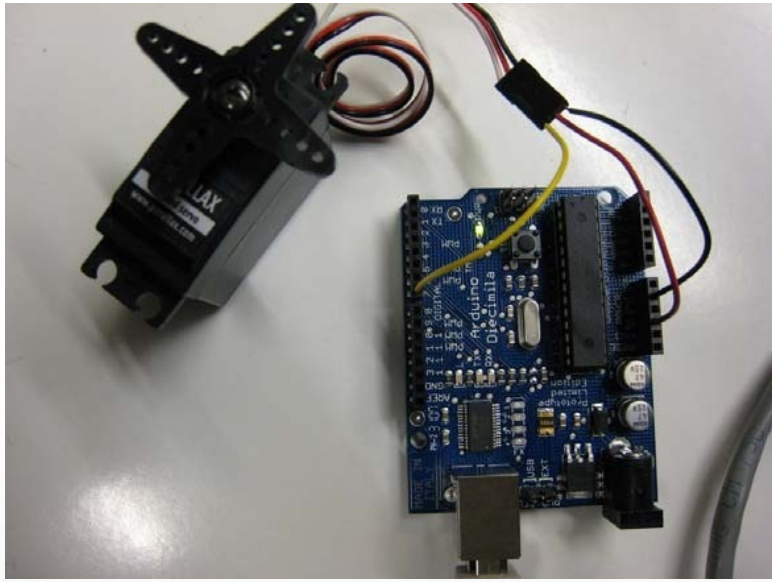


Servo Motors are electronic devices that convert digital signal to rotational movement. There are two sorts of servo motors: Standard servos that their rotation is limited to maximum of 180 degrees in each direction and Continuous Rotation Servos that can provide rotation unlimitedly in both directions

Arduino-DigitalOutput-Motion-Servo Motor

A servo motor is a motor that pulses at a certain rate moving its gear at a certain angle. It has three connections: **the black is ground, the red is connected to 5V, and the white (yellow wire here) is set to the digital pin.**





Arduino-Standard Servo Rotation to Exact Angel

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop()
{
  myservo.attach(9);
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  myservo.detach();                  //Detach the servo if you are not controling it for a while
  delay(2000);
}
```

Arduino-Standard Servo Rotation to Exact Angel

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop()
{
 myservo.attach(9);
 for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
 {                                  // in steps of 1 degree
  myservo.write(pos);               // tell servo to go to position in variable 'pos'
  delay(15);                        // waits 15ms for the servo to reach the position
 }
 for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
 {
  myservo.write(pos);               // tell servo to go to position in variable 'pos'
  delay(15);                        // waits 15ms for the servo to reach the position
 }
  myservo.detach();                 //Detach the servo if you are not controling it for a while
 delay(2000);
}
```

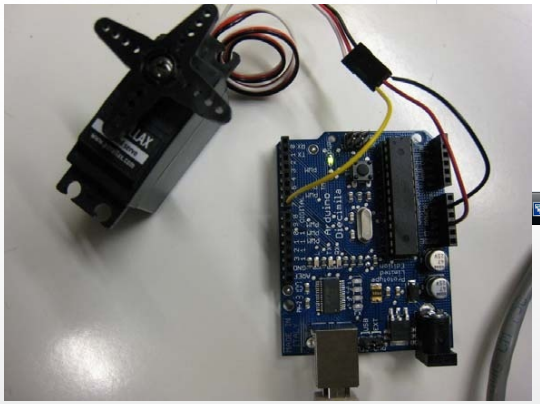
Arduino-Controlling Standard Servo with User Input

```
void setup(){
pinMode(5,OUTPUT); //set the pin to output
Serial.begin(9600); //open the serial to print
Serial.print("Ready"); //write a message
Serial.println();
}
void loop(){
int val = Serial.read(); //read the serial to see
if(val>='0' && val <= '9'){ //which key was pressed
val = val - '0'; //convert the character to an integer
val = val * (180/9); //9 divisions of 180 degrees
Serial.print("moving servo to ");
Serial.print(val);
Serial.println();
for(int a=0; a<100; a++){
int pulseWidth = (val*11)+500; // See the formula above
digitalWrite(5,HIGH);
delayMicroseconds(pulseWidth);
digitalWrite(5,LOW);
delay(10);
}
}
}
```

Arduino-Controlling Standard Servo with User Input

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
int angleValue=0;
int serialNumber=0;
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);
}
void loop()
{
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    myservo.attach(9);
    angleValue=serialNumber%180;
    myservo.write(angleValue);              // tell servo to go to position in variable 'pos'
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
    delay(250);
  }
  myservo.detach();
}
```

Arduino-Controlling Servo with User Input



```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
int angleValue=0;
int serialNumber=0;
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);
}
void loop()
{
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    myservo.attach(9);
    angleValue=serialNumber%180;
    myservo.write(angleValue);              // tell servo to go to position in variable 'pos'
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
    delay(250);
  }
  myservo.detach();
}
```

Arduino-DigitalOutput - Continuous Rotation

As opposed to standard Servo that its rotation is limited to 180 degrees both ways, a continuous rotation servo can keep rotating unlimitedly-again both ways- based on the frequency that is pulsed out to it. There is a specific frequency at which the Servo motor should be static and beyond and before which the servo will change in its rotation direction.