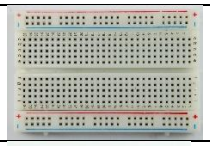
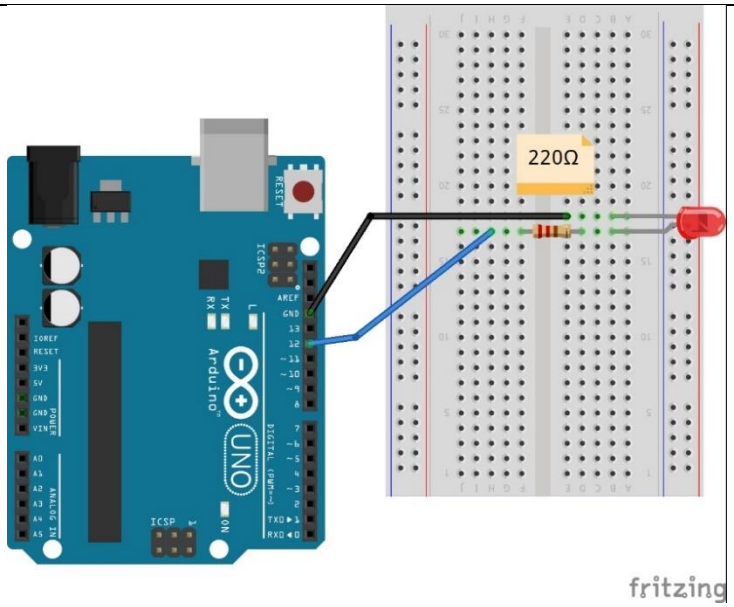
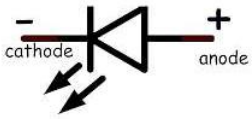
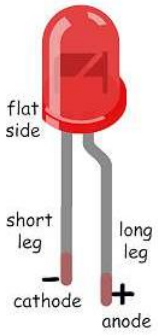
# Activity #1 Blinking Led

In this activity we will try to make a led flashing every second.

| **Needed Parts** | **Circuit wiring** |
|---|---|
| Breadboard  |   |
| Red Led  | |
| 220 or 330 Ohm Resistor  | |
| Jumper Wires  | |




Pin 12 controls the output pin, and a 220 Ohm resistor is connected between the output and the led, to protect led from a large amount of current. Usually LEDs have a longer leg to mark possitive connection (anode +). The other leg of the led is connected in the GND port od Arduino (cathode -).

***Programming Arduino***

```
// Blinking LED
void setup() {
        pinMode(12, OUTPUT);          // Pin12 is defined as OUTPUT.
}
 void loop() {                        // Repeated instructions infinitely (loop)
        digitalWrite(12, HIGH);               // 5 Volt (HIGH) is applied in Pin12
        delay(1000);                          // Wait 1 sec (1000msec)
        digitalWrite(12, LOW);                // 0 Volt (LOW) is applied in Pin12
        delay(1000);                          // Wait 1 sec (1000msec)
}
```

**Exercise #1**

**A.** Try to connect the OUTPUT to pin13 and make changes to the program so that the led flashes based on the new wiring.

**B.** Modify the program so that the led flashes following next instructions:

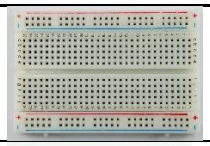
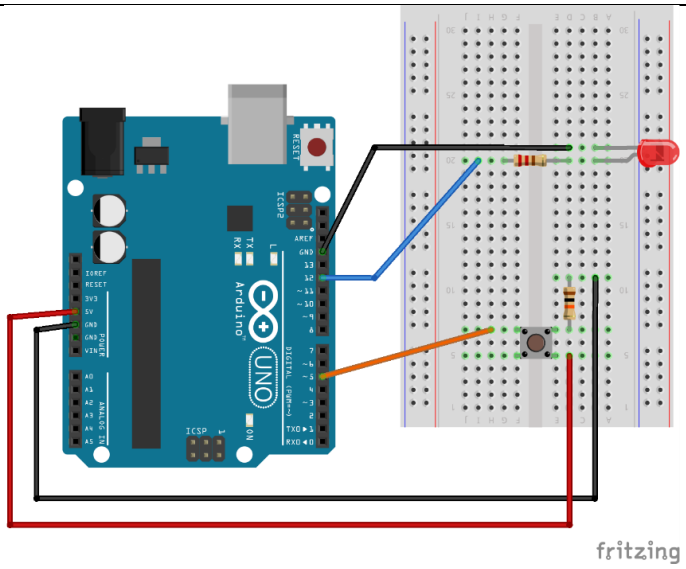
- initially, the led lights up for 3 seconds
- afterwards turns off and on 3 times every 1 second
- afterwards remains off for 2 seconds

**Solution for Exercise #1.B**

```
int i;                                  // counter variable
void setup() {
        pinMode(13, OUTPUT);            // Pin13 is defined as OUTPUT.
}
 void loop() {                          // Repeated instructions infinitely (loop)
        digitalWrite(13, HIGH);                 // 5 Volt (HIGH) is applied in Pin13
        delay(3000);                            // Wait 1 sec (1000msec)
        for (i=0; i<3; i++) {
                digitalWrite(13, LOW);                  // counter i used to repeat instructions 3 times
                delay(1000);
                digitalWrite(13, HIGH);
                delay(1000);
        }
        digitalWrite(13, LOW);
        delay(2000);
}
```

# Activity #2 Button Blinking Led

In this activity we will try to make a led flashing every time we push the button.

| Needed Parts | | Circuit wiring |
|---|---|---|
| Breadboard |  |  |
| Red Led |  | |
| 220 or 330 Ohm Resistor |  | |
| 10 kΩ Resistor |  | |
| Jumper Wires |  | |
| Push Button |  | |

We place an additional **push button** to control the input of the circuit. When the button is pressed, lets the power go on and turns on the LED. One pin of the button is connected to the **Pin 5** and the other pin to a a 10 kΩ resistor and then to the GND port, just to protect the button from overpower. The back pin of the button goes to the 5 V voltage of the Arduino.

***Programming Arduino***

Complete instructions needed (green colored).

```
// Button Blinking LED
int button=5;                          // Variable button equals 5
int _____                              // redLed=12;
void setup()    {
        pinMode(button , INPUT);       // Define Pin5 (through variable button) as INPUT.
_____                                  // pinMode(redLed , OUTPUT);}


void loop()     {
        int btnState = digitalRead(button);     // Read Digital Input (Pin5).
        if (btnState == 1)                      // If button is pressed
                _____                                   // digitalWrite( redLed , HIGH ); … Turn On Led
        else
                _____                                   // digitalWrite( redLed , LOW );… Turn Off Led
```

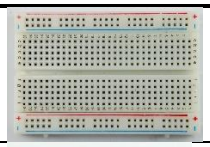
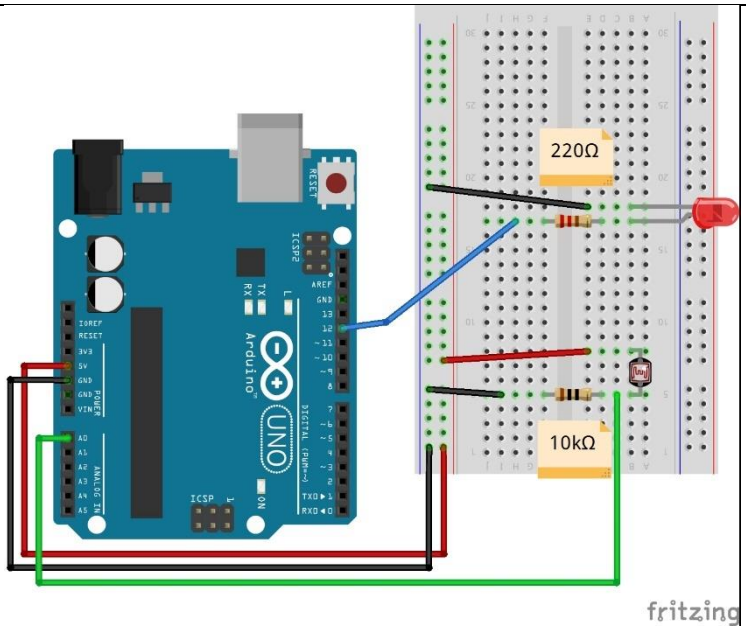
```
}
```

Another simpler solution is to use only one instruction:

```
void loop()       {
        digitalWrite( redLed , digitalRead(button));          // Turns On/Off Led, according to Digital Input state
}
```

# Activity #3 Photoresistor (LDR)

In this activity we will try to make a led flashing when there is not enough light in the room.

| Needed Parts | | Circuit wiring |
|---|---|---|
| Breadboard |  |   |
| Red Led |  | |
| 220 or 330 Ohm Resistor |  | |
| 10 kΩ Resistor |  | |
| Photoresistor |  | |
| Jumper Wires |  | |

We keep Led wiring from Activity #2, and we add a photoresistor connected to the analog **output port A0** of Arduino, through a 10kΩ resistor. The difference from previous connections is the use of Breadboard lines for GND grounding and 5 V voltage.

***Programming Arduino***

```
// Photoresistor lights LED
int redLed=12;                          // Variable redLed = 12

void setup() {
        Serial.begin(9600);             // Initialize serial port
        pinMode( redLed , OUTPUT);      // Define Pin12 as OUTPUT.
}

void loop()  {
        int sensorValue = analogRead(A0);   // Put Analog Input data from pin A0 to variable

        Serial.println(sensorValue);        // Print on monitor the collected data through serial port

        delay(200);                         // Delay 0,2 sec
}
```

The values of light level from the photoresistor are shown to the serial port monitor. If there is much light in the room the value is high (less resistance) otherwise value is reduced (more resistance). Cover the photoresistor with your hand and look on the data in serial port monitor.

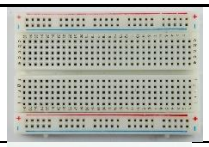
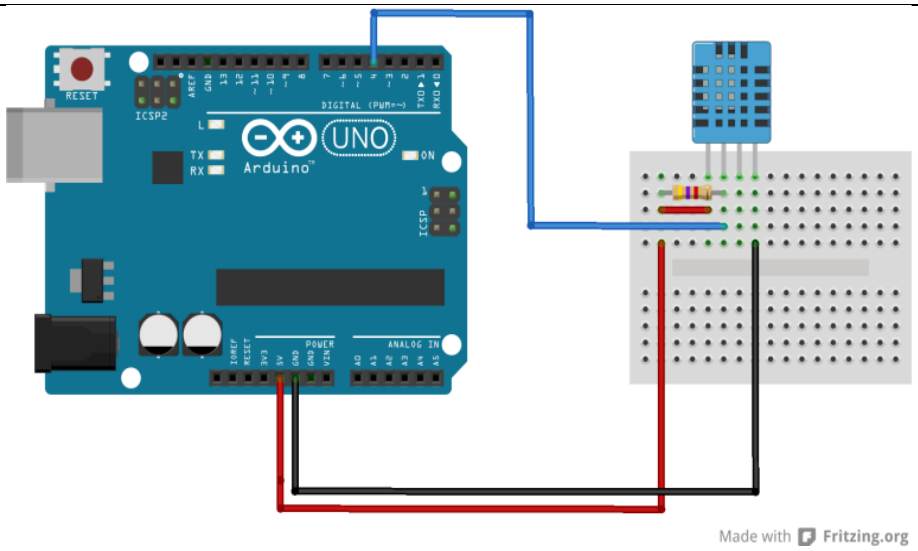
**Exercise #3**

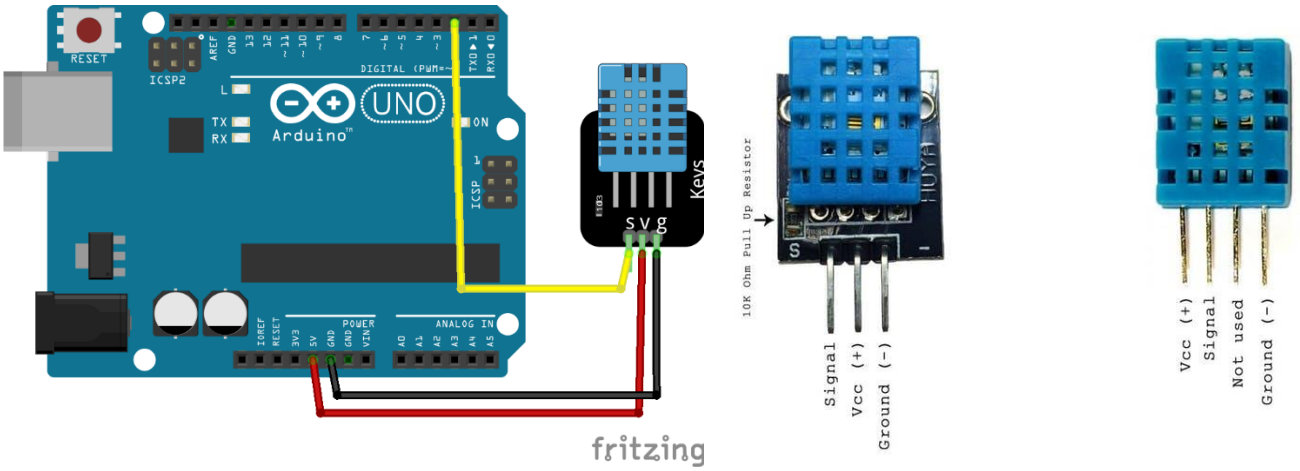
Complete the needed instructions to the previous code, to keep the LED off and turn it on whenever the light level gets less than 400.

```
if (sensorValue<400)
        digitalWrite(redLed, HIGH);
 else
        digitalWrite(ledLed, LOW);
```

# Activity #4 Temperature sensor (DHT11)

In this activity we will try to make a led turn on when the temperature is over a predefined limit.

| Needed Parts | | Circuit wiring |
|---|---|---|
| Breadboard |  |   |
| Red Led |  | |
| 220 or 330 Ohm Resistor |  | |
| 10 KOhm Resistor |  | |
| DHT11 (3 or 4 pins) |  | |
| Jumper Wires |  | If DHT11 has 3 pins and it's fitted on a board, there is no need to connect the resistor |




***Programming Arduino***

```
// DHT11 controls LED
#include <dht11.h>                    // Import sensor's library

#define DHT11PIN 4                    // Define variable to pin4

dht11 DHT11;


void setup() {
```

```
        Serial.begin(9600);                    // Initialize serial port
}
void loop() {
        Serial.println();                      // Print data to serial monitor
        int chk = DHT11.read(DHT11PIN);
        Serial.print("Humidity (%): ");
        Serial.println((float)DHT11.humidity, 2);
        Serial.print("Temperature (C): ");
        Serial.println((float)DHT11.temperature, 2);
        delay(2000);
}
```

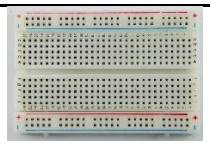
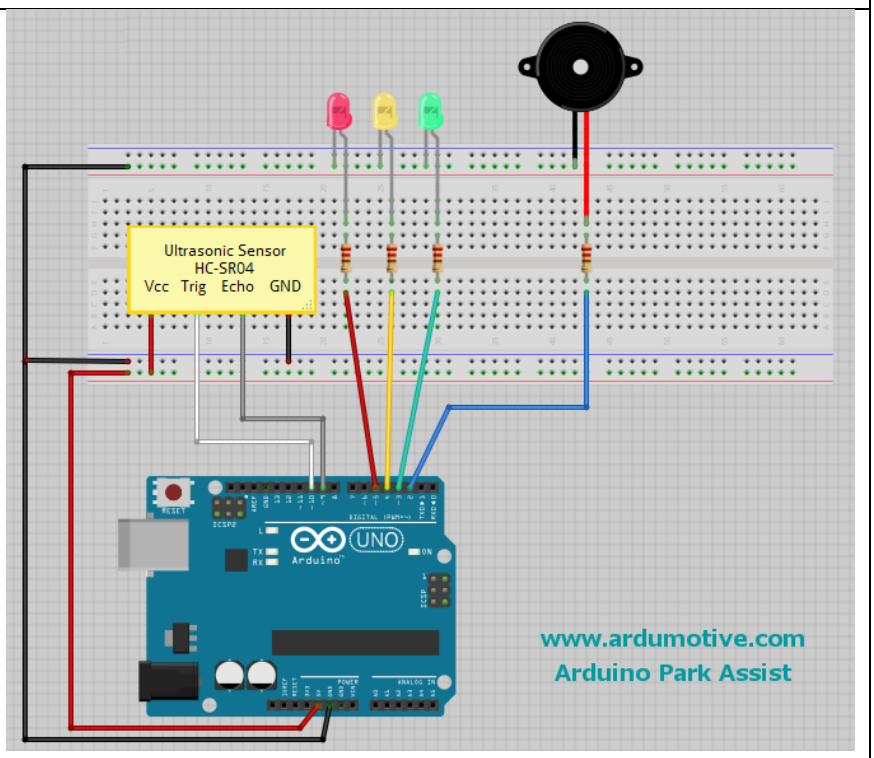
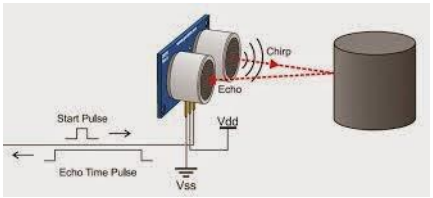
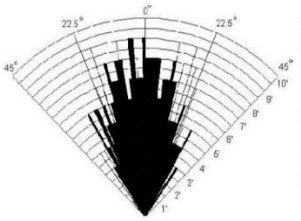
**Exercise #4**

Add the right instructions to the previous code, to keep the LED off and light it whenever the temperature is more than 25 degrees.

```
if (DHT.temperature >25)
                digitalWrite(13, HIGH);
        else
                digitalWrite(13, LOW);
```

# Activity #5 Ultrasonic sensor – Park assistant

In this activity an ultrasonic sensor is used to measure the distance of objects and to light the different Leds according to distance. Specifically, for up to 10 cm all leds turn on, for distance of 10 to 20 cm the yellow & green leds led turn, and for more than 20 cm only the green led turns on.

| **Τα υλικά που χρειάζονται** | | **Συνδεσμολογία των υλικών** |
|---|---|---|
| Breadboard |  |  |
| 1 x Red Led<br>1 x Yellow Led<br>1 x Green Led |  | |
| 3 x 220 or 330 Ohm Resistor |  | |
| Ultrasonic HC-SR04 |  | |
| Jumper Wires |  | |
| Piezo Buzzer |  |  <br>  |

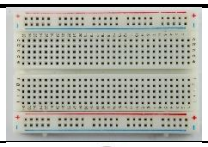
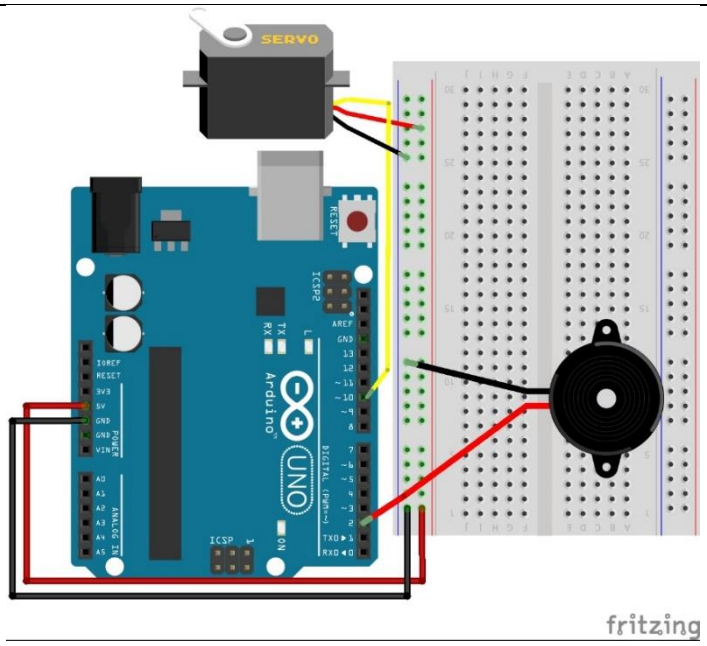
***Programming Arduino***

```
// Ultrasonic sensor controls leds
#include "Ultrasonic.h"                 // Import sensor's library
int buzzer = 2;                         // Buzzer at Pin 2
int greenLed = 3;                       // Green Led at Pin 3
int yellowLed= 4;                       // Yellow Led at Pin 4
int redLed = 5;                         // Red Led at Pin 5
int distance;                           // Declair variable distance
Ultrasonic ultrasonic(10,9);            //Trig (at Pin 10) and Echo (at Pin 9) of HC-SR04
```

```
void setup() {
 Serial.begin (9600);                    //Serial Port init at 9600
 pinMode(buzzer, OUTPUT);
 pinMode(greenLed, OUTPUT);              // Define OUTPUTs
 pinMode(yellowLed, OUTPUT);
 pinMode(redLed, OUTPUT);
}
void loop() {
 Serial.print(ultrasonic.Ranging(CM));          //Test your sensor in serial monitor
 Serial.println("cm");                          // Print distance in cm
 distance = ultrasonic.Ranging(CM);             //Take measurement
     if (distance >= 10 && distance <=20) {
      tone(buzzer, 5000, 100);                  // if distance between 10 to 20 cm turn on green led
      digitalWrite(greenLed, HIGH);
     }
     else if (distance >=5 && distance <10) {
      tone(buzzer, 5000, 200);                  // if distance over 5cm turn on green & yellow led
      digitalWrite(greenLed, HIGH);
      digitalWrite(yellowLed, HIGH);
     }
     else if (distance <5) {
      tone(buzzer, 5000, 500);
      digitalWrite(greenLed, HIGH);             // if distance less than 5cm turn on all leds
      digitalWrite(yellowLed, HIGH);
      digitalWrite(redLed, HIGH);
     }
 delay(500);
 digitalWrite(greenLed, LOW);
 digitalWrite(yellowLed, LOW);                  // distance more than 20cm leds turn off
 digitalWrite(redLed, LOW);
 noTone(buzzer);
}
```

# Activity #6 Servo + Buzzer

Servo motor rotates 90 degrees per second and makes different sounds.

| Needed Parts | | Circuit Wiring |
|---|---|---|
| Breadboard |  |   |
| Servo motor |  | |
| Potentiometer 10KOhm |  | |
| Jumper Wires |  | |

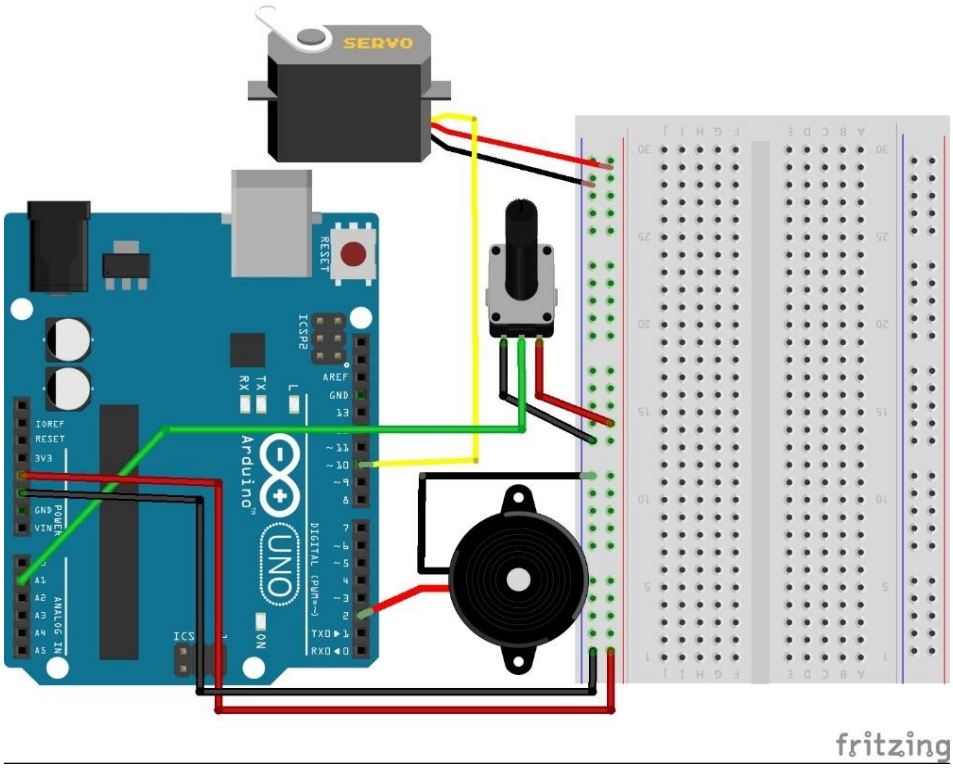
***Programming Arduino***

```
// Singing Servo
#include <Servo.h>                          //Add servo motor library
int buzzer = 2;                             // Variable buzzer to pin 2
int sPin = 10;                              // Variable sPin to pin 10
Servo s;                                    // Define variable s as servo type

void setup() {
        s.attach(sPin);                     // Correlation variable s with sPin servo for control
        pinMode(buzzer, OUTPUT);            // Define 2 (buzzer) as OUTPUT
}

void loop() {
        s.write(90);                        // Rotate servo 90 degrees
        tone(buzzer, 2000, 500);            // Sound 2000 Hz for 500 ms
        delay(1000);
        s.write(179);                       // Rotate servo 180 degrees
```

```
    tone(buzzer, 3500, 500);                // Sound 3500 Hz for 500 ms
    delay(1000);
    s.write(90);                            // Rotate servo back to 90 degrees
    tone(buzzer, 5000, 500);
    delay(1000);
    s.write(0);                             // Rotate servo back to 0 degrees
    noTone(buzzer);                         // No sound
    delay(1000);
}
```

## Exercise #6

Servo motor rotates from 0 to 180 degrees by using (turning) the potentiometer. We keep the previous wiring and we add a potentiometer to Analog port A1 of Arduino (see circuit image)




```
#include <Servo.h>                      //Add servo motor library
int buzzer = 2;                         // Variable buzzer to pin 2
int sPin = 10;                          // Variable sPin to pin 10
Servo s;                                // Define variable s as servo type
int potPin = A1;                        // Define variable potPin to pin A1
int potVal = 0;                         // Set potPin=0
int sVal = 0;                           // Set sVal=0
```

```
void setup() {
        Serial.begin(9600);
        s.attach(sPin);                    // Correlation variable s with sPin servo for control
        s.write(0);                        // Calibrate servo to 0 degrees
        pinMode(buzzer, OUTPUT);
        pinMode(potPin, INPUT);            // Potentiometer as INPUT
}

void loop() {
        potVal = analogRead(potPin);
        sVal = map(potVal, 0, 1023, 0, 179);
        s.write(sVal);
        tone(buzzer, sval*10, 500);
        Serial.println(sVal);
}
```

**Significant Note**

Often, we need to assign a value that belongs to a numbers range to another value that belongs to another number range. This math operation is relatively simple, but the Arduino provides us with a function to do this, the

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

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.

**For example:**
**val = map(val, 0, 1023, 0, 179);**

**/* we transform (remap) an analog value from the potentiometer (0 to 1023) to a value belonging to degrees for a servo (0 to 179) */**