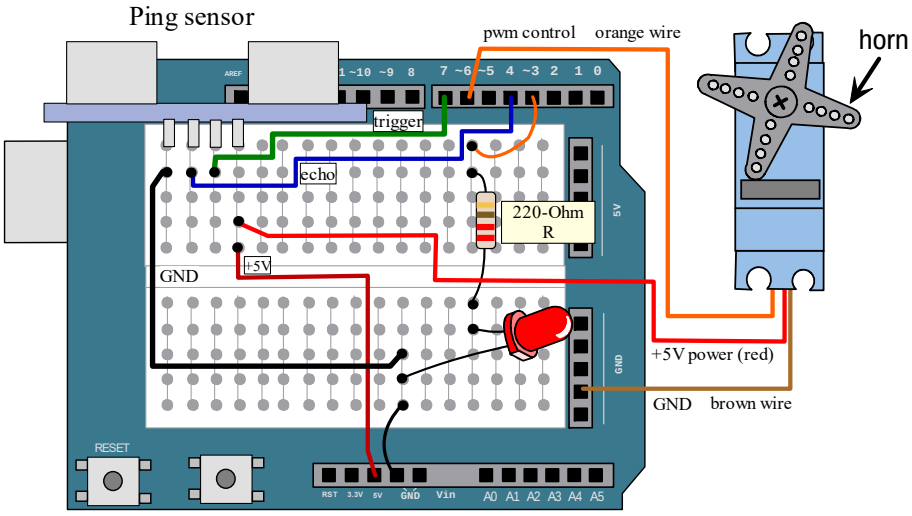
Electric Circuits Laboratory

ENGR 250L - LAB EXERCISES

LAB 10 - PING, LED, SERVO (.5 lec + 1h)



In a prior lab we have learned to control the lighting of an LED, and we have learned to collect distance data from a ultrasonic "ping" sensor. Now we'll learn to control the movement of a servo motor. After that we'll put things together by using the ping distance data to control the LED and the servo.

Using sensor data to control actuators is the basis for robotics and control systems.

PARTS LIST

1. Arduino, cable, computer
2. Breadboard shield
3. LED
4. 220-ohm R
4. ultrasonic sensor (HC-SR04)
5. micro servo motor
6. wire

10.1 SERVO MOTOR

REFER to CT GUIDE on servo motors.

Here we learn how to control a small servo motor. We will use Arduino's "servo" library.
We will do 3 servo exercises.

1. Servo to position - motor "servos" to a specified angle (try changing the angle)
2. Oscillations in steps - motor moves back and forth in steps
3. Oscillation as sine wave - motor oscillates smoothly according to a sine wave.

CIRCUIT & WIRING

Keep the prior circuit work - do NOT remove it.

But now ADD the servo wiring as shown.
Servo motors have 3 wires (power, GND, control signal). They are usually color-coded.
It is essential that you wire these correctly.

power - red
GND - black or brown
signal - orange, yellow, or white

10.1.1 Servo 1 - Servo to Position

CODE

```
// servo1_position.ino --------------------
// drive a servo motor to an angular position


#include <Servo.h>
Servo myservo;

int servoPin = 6;          //this # must match the circuit & pin must be pwm
int servoAngle = 90;

void setup()
{
myservo.attach(servoPin);
//myservo.writeMicroseconds(900);
myservo.write(servoAngle);
}

void loop()
{
}
```

OUTPUT

The motor should "servo" to the programmed angle. After that, it stays still. Try changing the servo angle to another value and re-upload. The servo should then move to the new angle and hold

10.1.2 Servo 2 - Step Oscillation

CODE

```
// servo2_stepOscill.ino --------------------
// Servo oscillates between 0 & 180 deg, in 20 deg increments (back and forth)

#include <Servo.h>
Servo myservo;

int servoPin = 6;

void setup()
{
  myservo.attach(servoPin);
}
```

```
void loop()
{
  //turn servo from 0 to 180 in 20 deg increments
  for (int i = 0; i <= 180 ; i=i+20)
  {
    myservo.write(i);
    delay(80);
  }

  //turn servo from 180 to 0 in 20 deg increments
  for (int i = 180; i >= 0 ; i=i-20)
  {
    myservo.write(i);
    delay(80);
  }
}
```

OUTPUT

The motor should oscillate back and forth in steps.

10.1.3 Servo 3 - Sine Wave Oscillation

CODE

```
// servo3_sin.ino --------------------
// servo oscillates using sin() function

#include <Servo.h>
Servo myservo;

int servoPin = 6;


void setup()
{
  myservo.attach(servoPin);
}


void loop()
{
  float freq = 0.5;   // in Hz
  float servoAngle;
  servoAngle = 90 + 80*sin(2 * PI * freq * millis()/1000.0);
  myservo.write(servoAngle);
}
```

OUTPUT

The motor should oscillate smoothly according to a sine wave.

10.2 PING-CONTROLLED LED

Here we will use the ping distance data to control the brightness of an LED.
The LED must be connected to a PWM pin.

CIRCUIT & WIRING

NO CHANGES from the prior exercise.
You should have the ping, LED, and servo connections/wiring in tact.

CODE

```
// ping2_distanceLED.ino ----------------
// use distance data to control LED brightness

const int trigPin = 7;
const int echoPin = 4;
const int ledPin = 3;           // use pwm pin

int ledLevel;

long duration;
int distance;
int distanceLast = 0;
int distanceCap = 400;

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  distance = getDistance(trigPin, echoPin);
  ledLevel = map(distance, 4, 30, 255, 0);
  ledLevel = constrain(ledLevel, 0, 255);

  analogWrite(ledPin, ledLevel);
  //Serial.print(distance);
  //Serial.print(ledLevel);
  //Serial.println(" ");
  delay(50);
}

int getDistance(int trigPin, int echoPin) // ---------------
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);      // send out 10 us pulse

  duration = pulseIn(echoPin, HIGH);   // get echo signal
  distance = duration * 0.0343 / 2;    // compute distance
```

```
  if (distance > distanceCap)   // if get bad data pt
  {
    distance = distanceLast;    // use last good data pt
  }
  else                          // good data pts
  {
    distance = constrain(distance, 2, 40);  // constrain distance: 2 - 40 cm
    distanceLast = distance;      // update last good value
  }

  return distance;
}

// end ---------------------------------------------------
```

OUTPUT

LED brightness will change with distance from ping sensor.

10.3 PING-CONTROLLED SERVO

Here we will use the ping distance data to control the angle of a servo motor.

CIRCUIT & WIRING

NO CHANGES from the prior exercise.
You should have the ping, LED, and servo connections/wiring in tact.

CODE

```
// ping3_distanceServo.ino ----------------
// use distance data to control servo angle

#include <Servo.h>
Servo myservo;

const int trigPin = 7;     // pins must match wiring
const int echoPin = 4;
const int ledPin = 3;
const int servoPin = 6;

int servoAngle = 90;

int ledLevel;

long duration;
int distance;
int distanceLast = 0;
int distanceCap = 400;

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
  pinMode(ledPin, OUTPUT);

  myservo.attach(servoPin);

  Serial.begin(9600);
}

void loop()
{
  distance = getDistance(trigPin, echoPin);
  ledLevel = map(distance, 4, 30, 255, 0);
  ledLevel = constrain(ledLevel, 0, 255);

  servoAngle = map(distance, 4, 30, 0, 180);
  servoAngle = constrain(servoAngle, 0, 180);

  analogWrite(ledPin, ledLevel);
  myservo.write(servoAngle);

  //Serial.print(distance);
  //Serial.print(ledLevel);
  //Serial.println(" ");
  delay(50);
}

int getDistance(int trigPin, int echoPin) // ---------------
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);      // send out 10 us pulse

  duration = pulseIn(echoPin, HIGH);  // get echo signal
  distance = duration * 0.0343 / 2;   // compute distance

  if (distance > distanceCap)   // if get bad data pt
  {
    distance = distanceLast;    // use last good data pt
  }
  else                          // good data pts
  {
    distance = constrain(distance, 2, 40);  // constrain distance: 2 - 40 cm
    distanceLast = distance;      // update last good value
  }

  return distance;
}

// end ----------------------------------------------------
```

OUTPUT

Servo angle will change with distance from ping sensor.