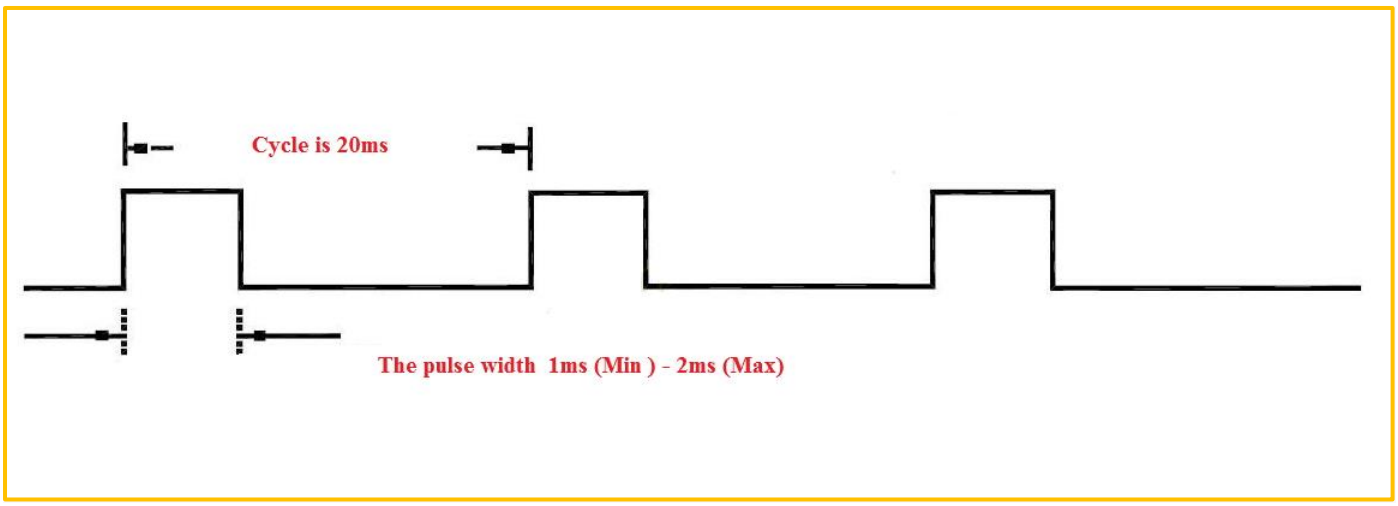
Introduction to Servo Motor

I guess you must have seen robots or high-tech products in American science-fiction films, at least heard of the noise of some moving automatic mechanical arms and audiences' scream. The noise comes from the rotation of the steering gear.

The steering gear is a kind of position (angle) servo driver, it can be rotated to any angle between 0 and 180 degrees, then precisely stop at your command, so it is suitable for those control systems which require angle changing and keeping . At present, it has been widely used in high-grade remote control toys, such as model aircraft, including the model plane, submarine model and remote control robot. Steering gear is an unprofessional name, in fact it is a kind of servo motor, a set of automatic control device which consists of DC motor, reduction gear group, sensor and control circuit. What is the automatic control? The so-called automatic control — continuously adjusting the output deviation by using a closed-loop feedback control circuit — makes the system output constant.

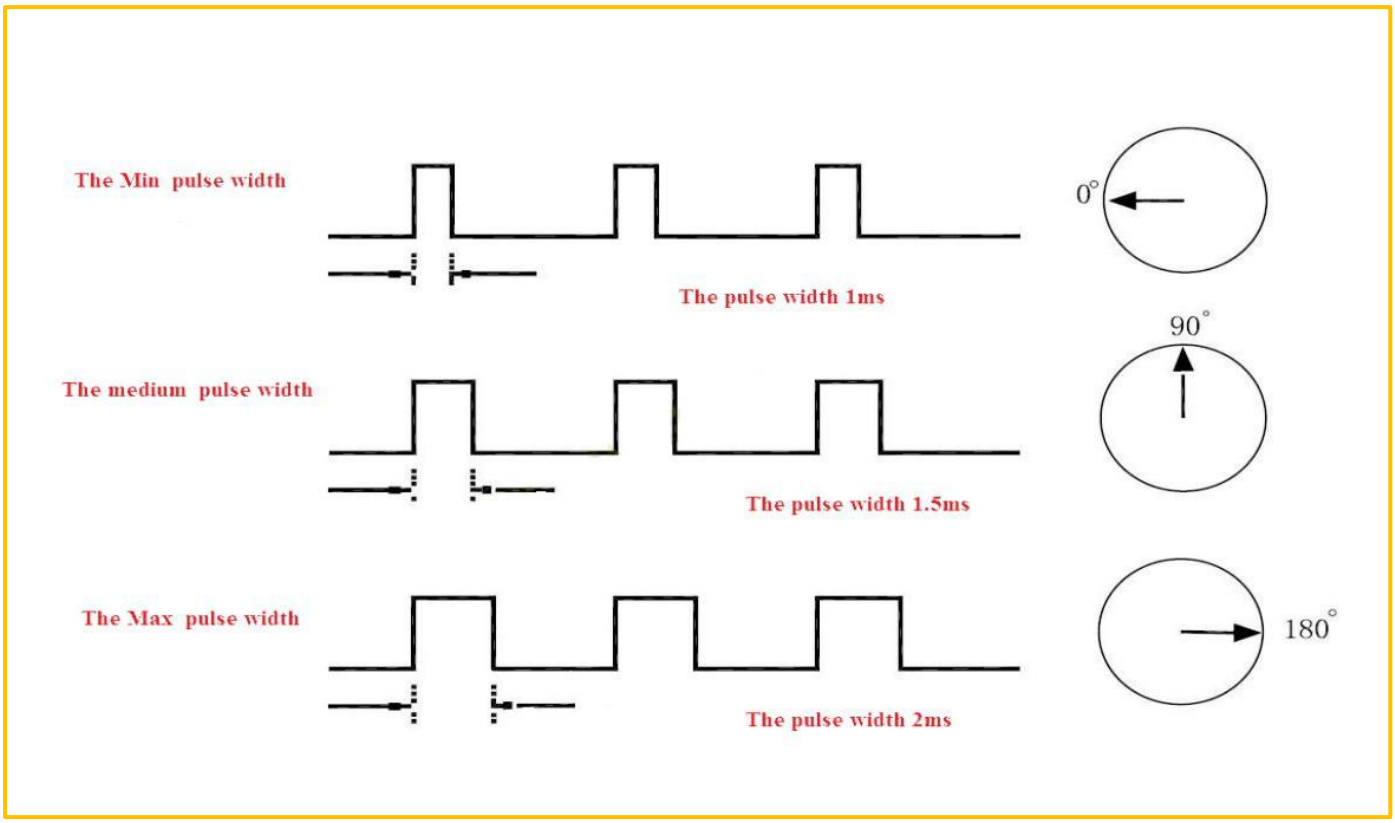
Experiment Principle

The steering gear servo system can be controlled by variable bandwidth pulse, the control line is used to transmit pulse. The parameters of the pulse consist minimum value, maximum value and frequency. In general, the cycle of the reference signal of the steering gear is 20ms, the bandwidth is 1.5ms. The reference signal is from the middle position. The steering gear has the maximum rotation angle, the middle position refers to the volumes from this position to the minimum angle and the maximum angle are exactly identical. The most important part, the maximum rotation angle varies with different steering gears, but of which the bandwidth of the middle position is certain, that is 1.5 ms.

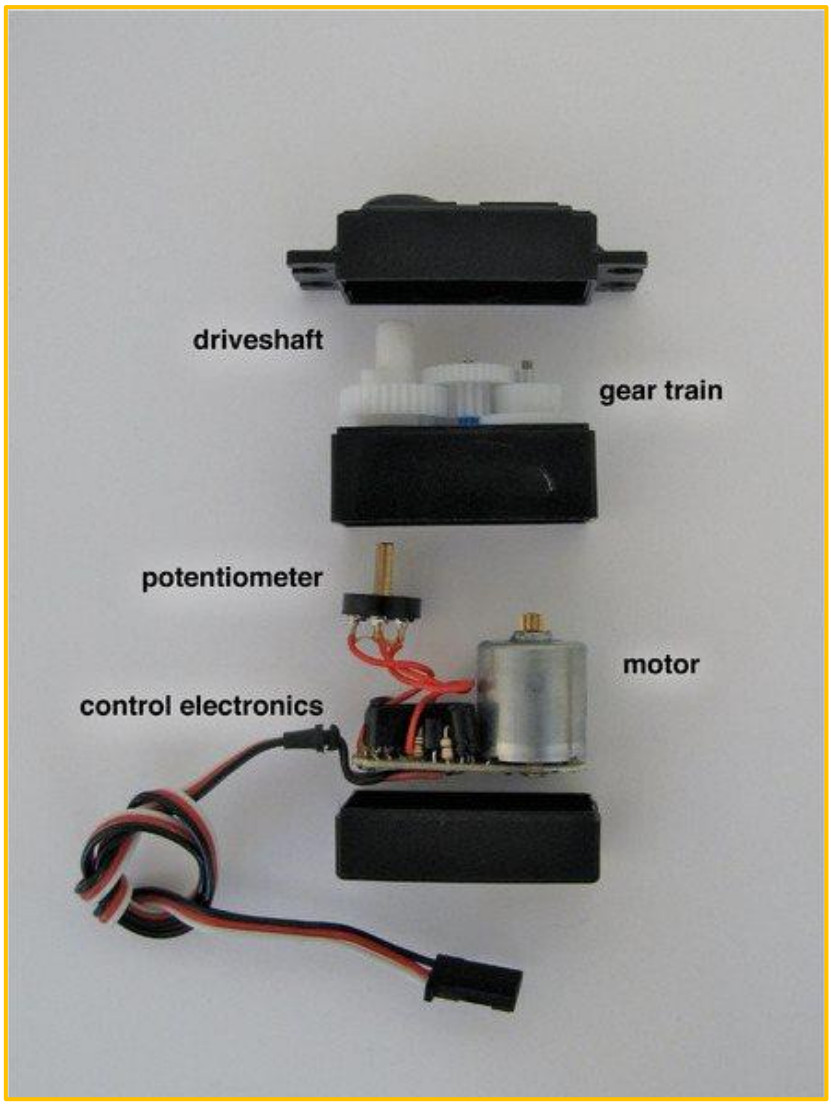


The rotation angle is produced by the continuous pulse from control line. This method is called pulse modulation. The length of pulse decides the rotation angle of steering gear. For example: the steering gear rotates to the middle position by 1.5 millisecond pulse(for 180 ° steering gear, the middle position is 90). When the control system issues commands to move the steering gear to a particular position and make it keep a certain angle, then the influence of the external force won't change the angle, but the ceiling is its biggest torsion. Unless the control system continuously issues pulse to stable the steering angle, the angle will not always stay the same.

When the steering gear receives a pulse less than 1.5ms , the output shaft will take the middle position as standard and rotate a certain angle counterclockwise; when the received pulse is greater than 1.5ms, then the output shaft rotates clockwise. Different brands of steering gears, and even the same brand of different steering gears, the maximum and minimum value could be different.



Internal Structure of Steering Gear



Experimental principle

Rduino is a convenient and flexible open source electronic prototype platform.After data is generated, it is transmitted back to the computer by serial port, and Processing is used to display the "radar" window. Processing originates from data visualization, and a lot of multimedia Processing capabilities have been added through evolution, so it can be expressed not only by graphics, but also by Processing video and sound.

Experiment purpose

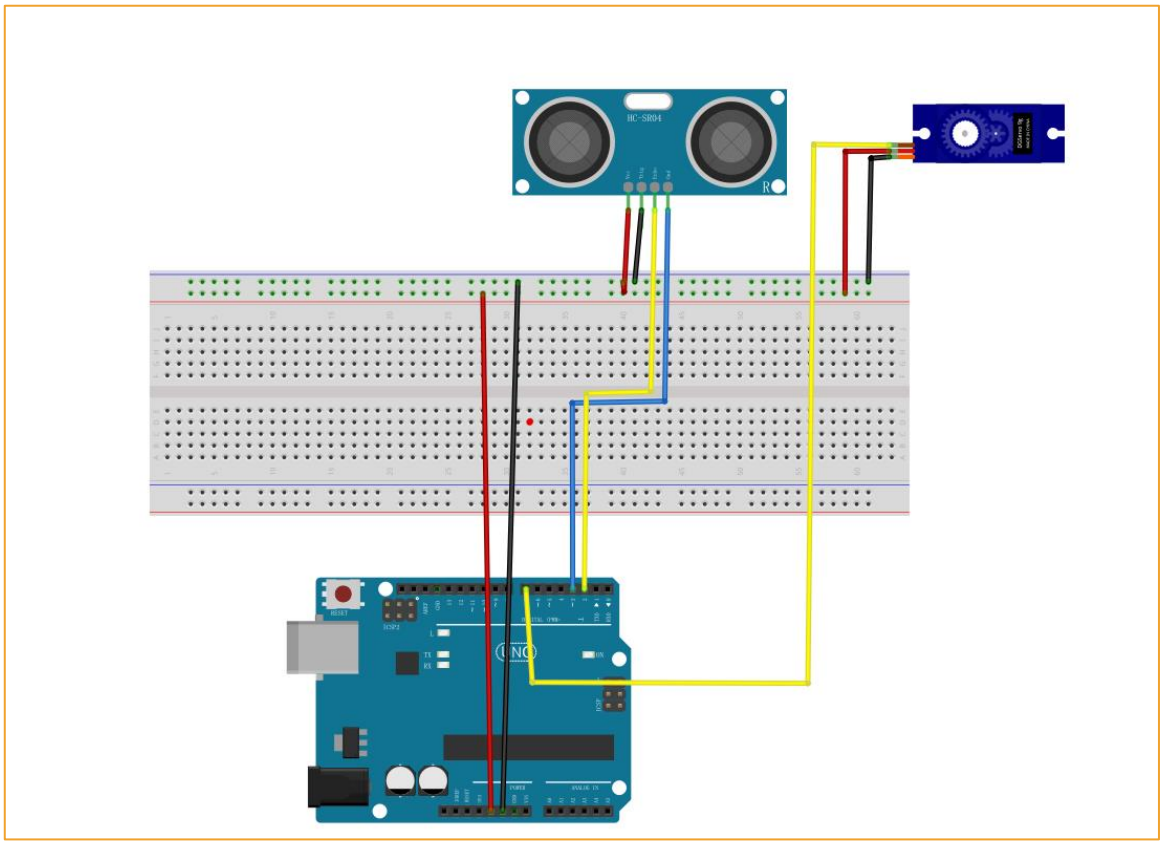
- Learn the control principle of steering gear;
- Learning ultrasonic ranging method;
- Ultrasonic ranging module is used to realize the function of ultrasonic radar.

The component list

- Arduinos Uno motherboard
- Bread plate
- USB cable
- Servo Motor*1
- Jumpers
- Ultrasonic module
- Steering gear bracket

Wiring

Arduino UNO	Servo motor
GND	GND
5V	VCC
7	S
Arduino UNO	RGB Ultrasonic module
GNG	GND
3	IO
2	RGB
5V	VCC



Code

```
#include<Servo.h>
const int SignalPin = 3;
const int motorSignalPin = 7;
const int startingAngle = 15;
const int minimumAngle = 15;
const int maximumAngle = 165;
const int rotationSpeed = 1;
Servo motor;
void setup(void)
{
    motor.attach(motorSignalPin);
    Serial.begin(9600);
}
void loop(void)
{
    static int motorAngle = startingAngle;
    static int motorRotateAmount = rotationSpeed;
    motor.write(motorAngle);
    delay(30);
```

```
    SerialOutput(motorAngle, CalculateDistance());
    motorAngle += motorRotateAmount;
    if(motorAngle <= minimumAngle || motorAngle >= maximumAngle) {
        motorRotateAmount = -motorRotateAmount;
    }}
int CalculateDistance(void)
{
    float distance;
    pinMode(SignalPin, OUTPUT);
    digitalWrite(SignalPin, LOW);
    delayMicroseconds(2);
    digitalWrite(SignalPin, HIGH);
    delayMicroseconds(20);
    digitalWrite(SignalPin, LOW);
    pinMode(SignalPin, INPUT);
    Time_Echo_us = pulseIn(SignalPin, HIGH);
    if ((Time_Echo_us < 60000) && (Time_Echo_us > 1)) {
        distance = Time_Echo_us / 58.00;
        Serial.print("distance is :");
        Serial.print(distance);
        Serial.print("cm");
        Serial.println();
    }
    return int(distance);
}
void SerialOutput(const int angle, const int distance)
{

 String angleString = String(angle);
    String distanceString = String(distance);
    Serial.println(angleString + "," + distanceString);
}
```

Processing Code

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial  myPort;
PFont orcFont;
int iAngle;
```

```
int iDistance;
void setup() {
size(1350, 760);
        smooth();

        myPort  =  new Serial(this, "COM31", 9600);
        myPort.clear();
        myPort.bufferUntil(&apos;\n&apos;);
            orcFont  =  loadFont("OCRAExtended-48.vlw");
}
void draw()
{
        fill(98, 245, 31);
        textFont(orcFont);
        noStroke();
        fill(0, 4);
        rect(0, 0, width, 0.935 * height);
        fill(98, 245, 31);

        DrawRadar();
        DrawLine();
        DrawObject();
        DrawText();
}
void serialEvent (Serial myPort)
{
        try {
                String data = myPort.readStringUntil(&apos;\n&apos;);
                        if (data == null) {
                        return;
                }
                  int commaIndex = data.indexOf(",");
                String angle = data.substring(0, commaIndex);
                String distance = data.substring(commaIndex+1, data.length()-1);
      iAngle  =  StringToInt(angle);
                iDistance  =  StringToInt(distance);
        }  catch(RuntimeException e) {}
}
void DrawRadar()
{
        pushMatrix();
        translate(width/2, 0.926 * height);
```

```
        noFill();
        strokeWeight(2);
        stroke(98, 245, 31);

        // draws the arc lines
        DrawRadarArcLine(0.9375);
        DrawRadarArcLine(0.7300);
        DrawRadarArcLine(0.5210);
        DrawRadarArcLine(0.3130);

        // draws the angle lines
        final int halfWidth = width/2;
        line(-halfWidth, 0, halfWidth, 0);
        for(int angle = 30; angle <= 150; angle+=30) {
                DrawRadarAngledLine(angle);
        }
        line(-halfWidth * cos(radians(30)), 0, halfWidth, 0);
                popMatrix();
}
void DrawRadarArcLine(final float coefficient)
{
        arc(0, 0, coefficient * width, coefficient * width, PI, TWO_PI);
}
void DrawRadarAngledLine(final int angle){
        line(0, 0, (-width/2) * cos(radians(angle)), (-width/2) * sin(radians(angle)));
}
void DrawObject()
{
        pushMatrix();
        translate(width/2, 0.926 * height);
        strokeWeight(9);
        stroke(255,10,10);
        int pixsDistance = int(iDistance * 0.020835 * height);
        if(iDistance < 40 && iDistance != 0) {
                float cos = cos(radians(iAngle));
                float sin = sin(radians(iAngle));
                int x1 = +int(pixsDistance * cos);
                int y1 = -int(pixsDistance * sin);
                int x2 = +int(0.495 * width * cos);
                int y2 = -int(0.495 * width * sin);

                line(x1, y1, x2, y2);
```

```
        }
            popMatrix();
}
void DrawLine()
{
        pushMatrix();
        strokeWeight(9);
        stroke(30, 250, 60);
        translate(width/2, 0.926 * height);

        float angle = radians(iAngle);
        int x = int(+0.88 * height * cos(angle));
        int y = int(-0.88 * height * sin(angle));
        line(0, 0, x, y);
        popMatrix();
}
void DrawText()
{
        pushMatrix();
        fill(0, 0, 0);
        noStroke();
        rect(0, 0.9352 * height, width, height);
        fill(98, 245, 31);
        textSize(25);
        text("10cm", 0.6146 * width, 0.9167 * height);
        text("20cm", 0.7190 * width, 0.9167 * height);
        text("30cm", 0.8230 * width, 0.9167 * height);
        text("40cm", 0.9271 * width, 0.9167 * height);

        textSize(40);
        text("Object: " + (iDistance > 40 ? "Out of Range" : "In Range"), 0.125 * width,
0.9723 * height);
        text("Angle: " + iAngle + " ° ", 0.52 * width, 0.9723 * height);
        text("Distance: ", 0.74 * width, 0.9723 * height);
        if(iDistance < 40) {
                text("        " + iDistance +" cm", 0.775 * width, 0.9723 * height);
        }
          textSize(25);
        fill(98, 245, 60);
        translate(0.5006 * width + width/2 * cos(radians(30)), 0.9093 * height - width/2
* sin(radians(30)));
        rotate(-radians(-60));
```

```
        text("30° ",0,0);

        resetMatrix();

        translate(0.497 * width + width/2 * cos(radians(60)), 0.9112 * height - width/2 *
sin(radians(60)));
        rotate(-radians(-30));
        text("60° ",0,0);
          resetMatrix();
          translate(0.493 * width + width/2 * cos(radians(90)), 0.9167 * height - width/2
* sin(radians(90)));
        rotate(radians(0));
        text("90° ",0,0);
        resetMatrix();

        translate(0.487 * width + width/2 * cos(radians(120)), 0.92871 * height - width/2
* sin(radians(120)));
        rotate(radians(-30));
        text("120° ",0,0);
        resetMatrix();

        translate(0.4896 * width + width/2 * cos(radians(150)), 0.9426 * height - width/2
* sin(radians(150)));
        rotate(radians(-60));
        text("150° ",0,0);
        popMatrix();
}

int StringToInt(String string)
{
        int value = 0;
        for(int i = 0; i < string.length(); ++i) {
                if(string.charAt(i) >= &apos;0&apos; && string.charAt(i) <= &apos;9&apos;)
{
                        value  *=  10;
                        value  +=  (string.charAt(i) - &apos;0&apos;);
                }}
          return value;}
```

Software operation steps

In this experiment, we need to use Processing software, Processing software download address:

https://processing.org/download/

We can download the corresponding version according to our computer system model

1) install the Processing software and turn on the program of ultrasonic radar experiment \Processing\sketch_161004a\ sketch_161004a.pde

2) turn on the ultrasonic radar experiment \Arduino_Radar\Arduino_Radar. Ino program with Arduino IDE 1.8.9

3) burn the arduino_radar. ino program to the Arduino UNO board

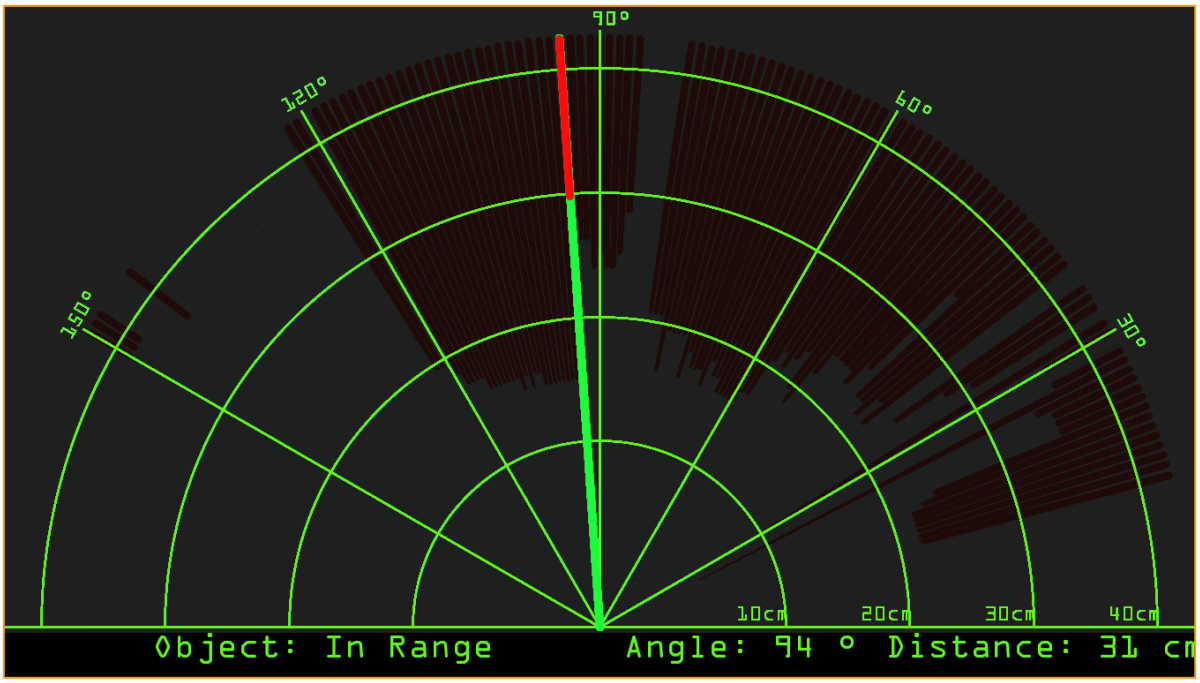
4) open the serial port monitor to observe the measurement data of ultrasonic ranging module printed by the serial port monitor

Check the port number of Arduino UNO, say COM31;MyPort = new Serial(this, "COM31", 9600);The port parameter of this function is changed to COM31, which is consistent with the Arduino port, otherwise an error will be reported

```
sketch_161004a
1  import processing.serial.*;
2  import java.awt.event.KeyEvent;
3  import java.io.IOException;
4
5  Serial myPort;
6  PFont orcFont;
7  int iAngle;
8  int iDistance;
9  void setup() {
10 size(1350, 760);
11     smooth();
12
13     myPort = new Serial(this, "COM31", 9600);
14     myPort.clear();
15     myPort.bufferUntil('\n');
16       orcFont = loadFont("OCRAExtended-48.vlw");
17 }
18 void draw()
```

6) click the run button of processing, as shown in the figure below;

7) after normal communication, processing software will show the interface shown in the figure below, and then the ultrasonic radar will start to work;


90°
120°
60°
150°
30°
10cm
20cm
30cm
40cm
Object: In Range
Angle: 94 °
Distance: 31 cm

Experiment Result