RGB ультразвуковой радиолокационный эксперимент

Цель эксперимента

- Изучить принцип управления рулевого механизма;
- Изучение метода ультразвукового ранжирования RGB;
- Используйте модуль ультразвукового определения дальности RGB для реализации функции ультразвукового радара.

Компоненты

- Материнская плата Keywish Arduino UNO R3
- Макетная плата
- USB-кабель для передачи данных
- SG90 сервопривод
- RGB ультразвуковой модуль
- Кронштейн сервопривода
- Несколько перемычек

Принцип работы рулевого механизма

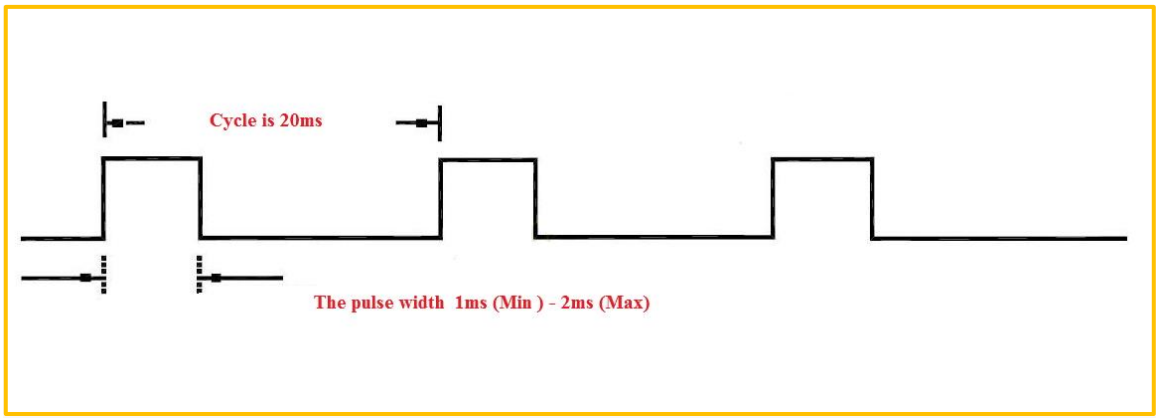
Сигнал управления сервоприводом поступает в микросхему модуляции сигнала из канала приемника для получения напряжения смещения постоянного тока. Он имеет опорную цепь внутри, генерирует опорный сигнал с периодом 20 мс и шириной 1,5 мс, сравнивает полученное напряжение смещения постоянного тока с напряжением потенциометра и получает выход разности напряжений. Наконец, положительный и отрицательный значения разности напряжений выводятся на микросхему привода двигателя для определения положительного и отрицательного вращения двигателя. Когда скорость двигателя постоянна, потенциометр приводится во вращение через каскадный редуктор, так что разность напряжений равна 0, и двигатель перестает вращаться. Рулевой механизм имеет максимальный угол поворота, промежуточное положение относится к объему от этого положения до минимального угла, а максимальный угол точно такой же. Самая важная часть, максимальный угол поворота изменяется в зависимости от различных рулевых механизмов, но определяется ширина полосы промежуточного положения, которая составляет 1,5 миллисекунды.

Управление рулевого механизма

Для управления рулевым механизмом обычно требуется базовый импульс времени, составляющий около 20 мс, а высокоуровневая часть импульса обычно является частью импульса управления углом в диапазоне от 0,5 мс до 2,5 мс. В качестве

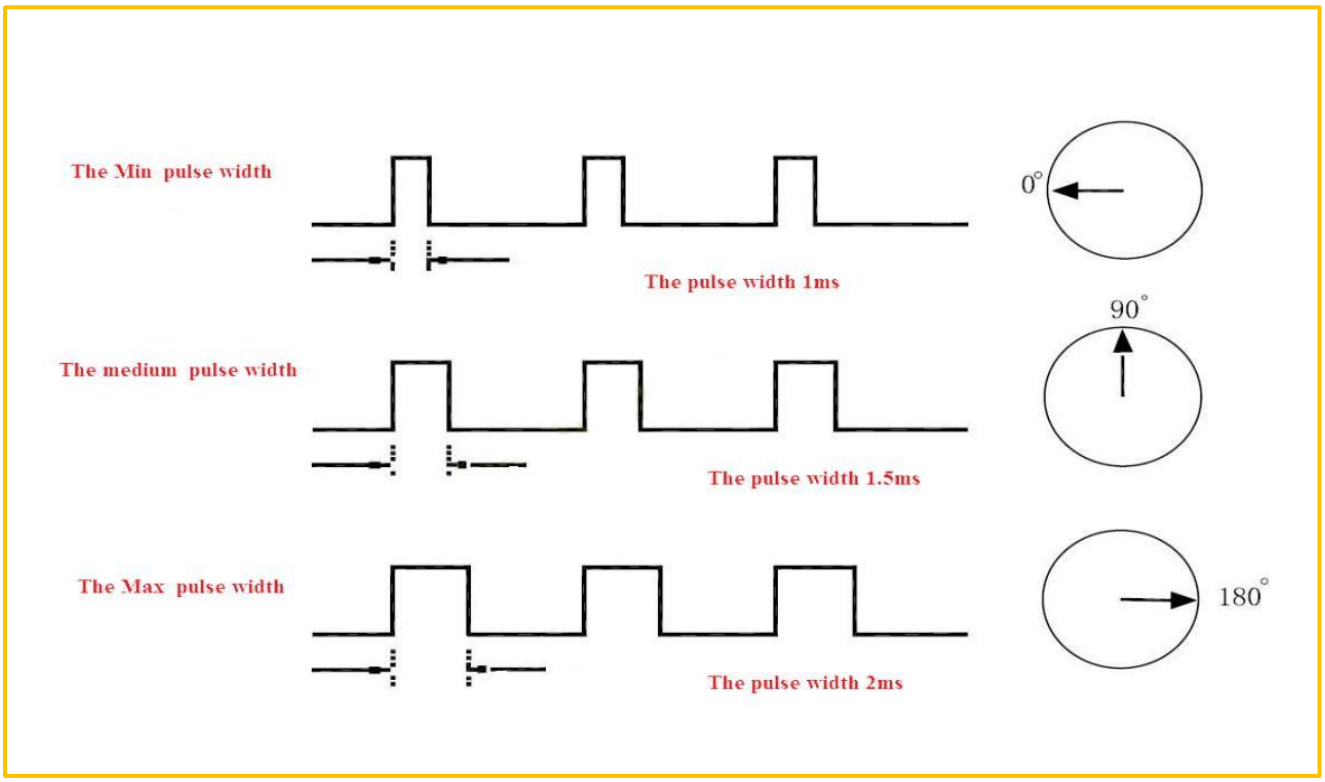
примера возьмем сервопривод угла на 180 градусов, затем соответствующие отношения управления будут следующими:

- 0,5 мс ---------- 0 градусов;
- Ms 1,0 мс -------- 45 градусов;
- Ms 1,5 мс -------- 90 градусов;
- 2.0 мс ---------- 135 градусов;
- 2,5 мс-------- 180 градусов;

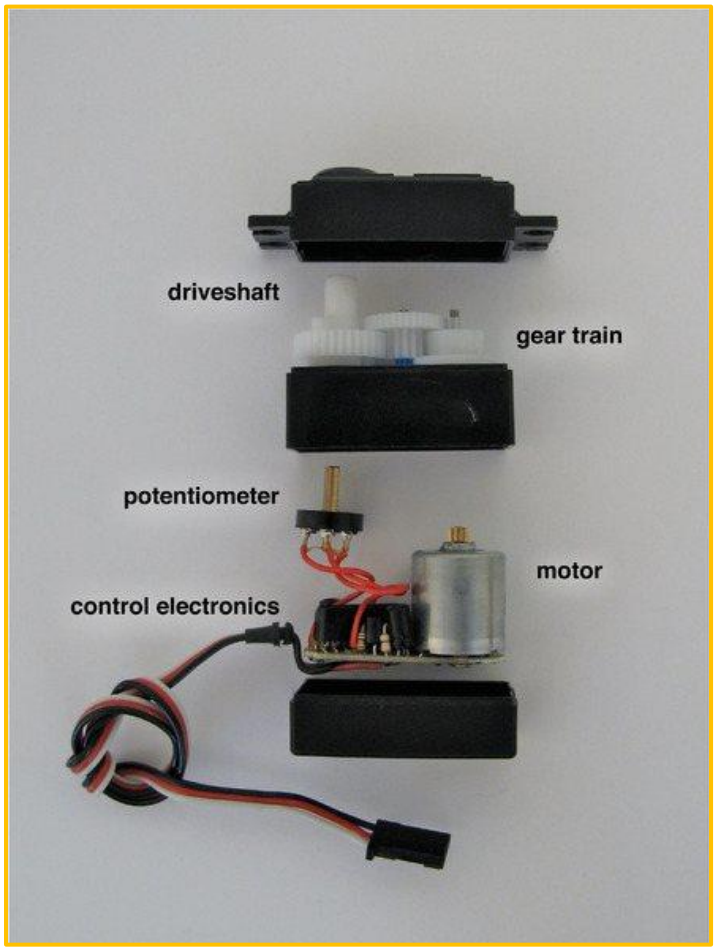


Угол поворота генерируется непрерывными импульсами от линии управления. Этот метод называется импульсной модуляцией. Длина импульса определяет угол поворота рулевого механизма. Например: сервопривод вращается в среднее положение в течение 1,5 мс импульса (для сервопривода 180 ° среднее положение составляет 90 °). Когда система управления выдает команду на перемещение рулевого механизма в определенное положение и удержание его под определенным углом, воздействие внешних сил не изменит этот угол. Если система управления не будет непрерывно пульсировать для стабилизации угла поворота рулевого колеса, угол не всегда останется неизменным

Когда сервопривод получает импульс менее 1,5 мс, выходной вал будет использоваться в качестве стандартного промежуточного положения, вращающегося против часовой стрелки на определенный угол, а при получении импульса, превышающего 1,5 мс, выходной вал будет вращаться по часовой стрелке. Различные марки рулевых механизмов, даже разные рулевые механизмы одной и той же марки, могут иметь разные максимальные и минимальные значения.



Внутренняя структура рулевого механизма



Принцип работы ультразвукового дальномера

Ультразвуковой передатчик RGB излучает ультразвуковые волны в определенном направлении, и мы одновременно начинаем синхронизацию. Когда ультразвуковая волна в воздухе сталкивается с препятствием, она немедленно возвращается, ультразвуковой приемник RGB принимает отраженную волну, и затем мы останавливаем синхронизацию. Скорость звуковой волны в воздухе составляет 340 м / с. По зарегистрированному времени t можно рассчитать расстояние s между начальной точкой и препятствием, а именно: s = 340 м / с * t / 2. Таким образом, мы можем получить расстояние.

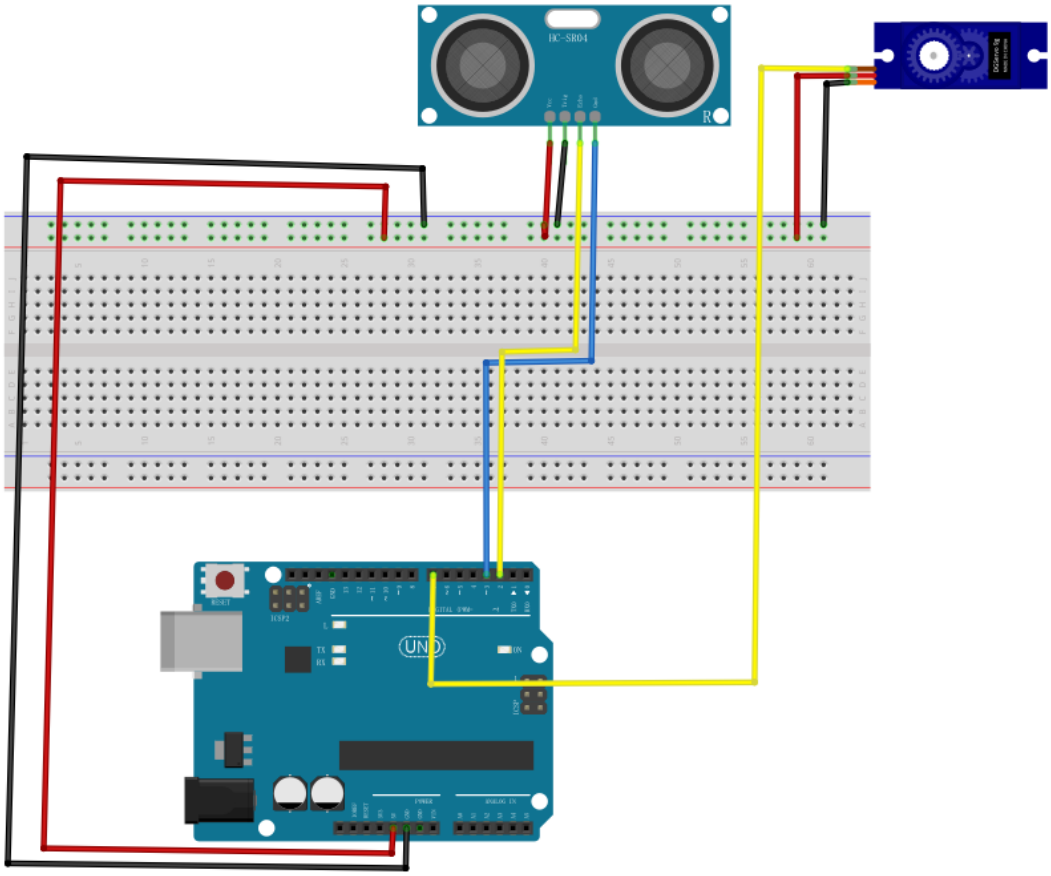
Ультразвуковой дальномер RGB имеет четыре контакта: Vcc, GND, RGB и IO - триггерные выводы для измерения расстояния. Пока контакт IO имеет высокий уровень не менее 20 мкс, модуль ультразвуковой передачи автоматически отправляет восемь ультразвуковых импульсов 40 кГц И автоматически определяет наличие сигнала возврата. Этот шаг будет автоматически выполнен внутренним модулем. Если есть какой-либо обратный сигнал, вывод IO будет выводить высокий уровень, длительность высокого уровня - это время от передачи ультразвуковой волны до возврата. В настоящее время мы можем использовать функцию pulseIn () для получения результата измерения расстояния и вычисления фактического расстояния.

Экспериментальный принцип

Arduino - это удобная и гибкая платформа с открытым исходным кодом для электронных прототипов. После того, как данные сгенерированы, они передаются обратно на компьютер через последовательный порт, а затем с помощью обработки для отображения окна «радара». Обработка происходила из визуализации данных. После эволюции было добавлено множество возможностей обработки мультимедиа. Поэтому он может не только выражать их с помощью графики, но также и видео и звук.

Проводка

Arduino UNO плата	**Рулевой механизм**
GND	GND
5V	VCC
7	S
Arduino UNO плата	**RGB ультразвуковой модуль**
GNG	GND
3	IO
2	RGB
5V	VCC


HC-SR04

Программа

```
#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 программа

```
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;}
```

Этапы работы программного обеспечения

В этом эксперименте нам нужно использовать программное обеспечение для обработки, адрес загрузки программного обеспечения для обработки: https://processing.org/download/Мы можем скачать соответствующую версию в соответствии с нашей моделью компьютерной системы

1) Установите программное обеспечение для обработки и откройте эксперимент с ультразвуковым радаром \ Processing \ sketch_161004a \ sketch_161004a.pde

2) Используйте программное обеспечение Arduino IDE, чтобы открыть ультразвуковой радарный эксперимент \ Arduino_Radar \ Arduino_Radar.ino.

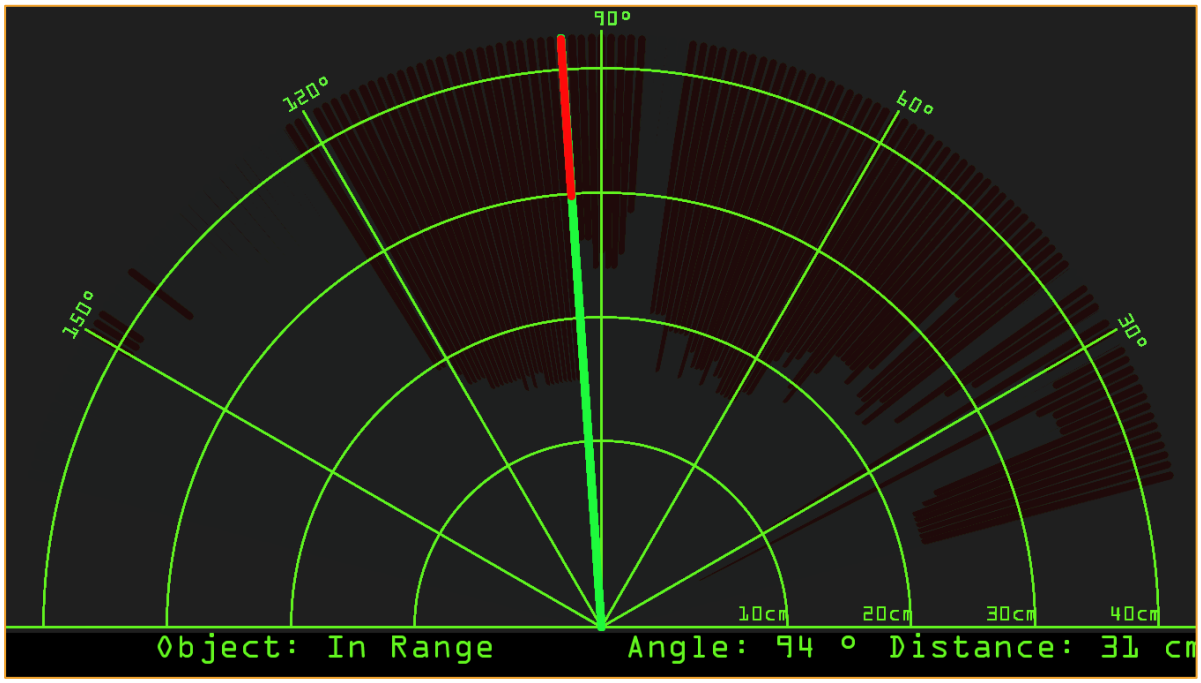
3) Запишите программу Arduino_Radar.ino на доску Arduino UNO

4) Откройте последовательный монитор и просмотрите данные измерений модуля ультразвукового измерения, напечатанные на последовательном мониторе.

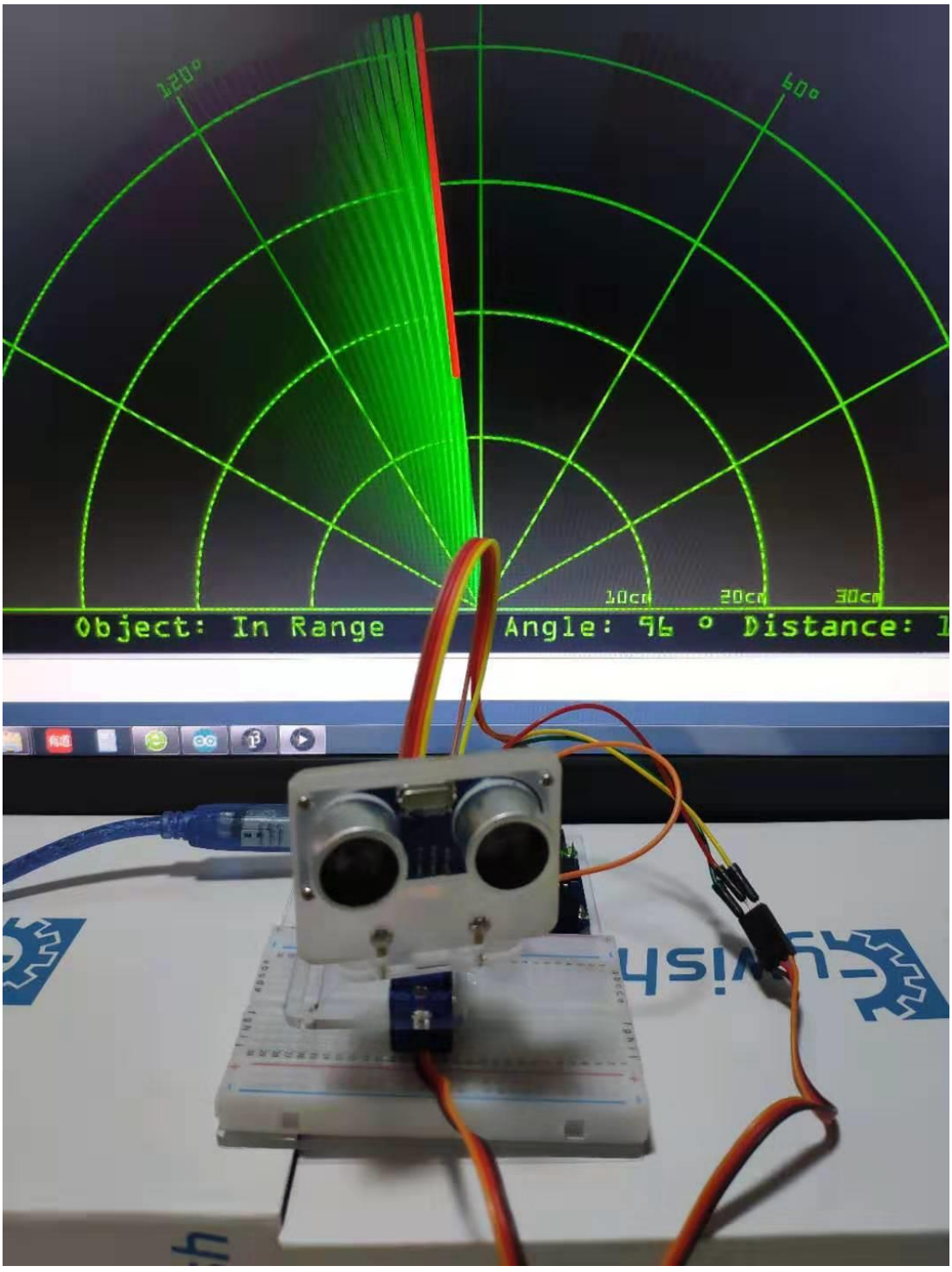
5) Проверьте номер порта Arduino UNO, если это COM31, измените myPort = new Serial (this, «COM31», 9600) в программе обработки, измените параметр порта этой функции на COM31, что соответствует порту Arduino, в противном случае он будет Сообщить об ошибке

1) Нажмите кнопку запуска обработки, как показано ниже;

2) После того, как связь нормальная, программное обеспечение обработки processing отобразит интерфейс, как показано ниже, и ультразвуковой радар начнет работать;

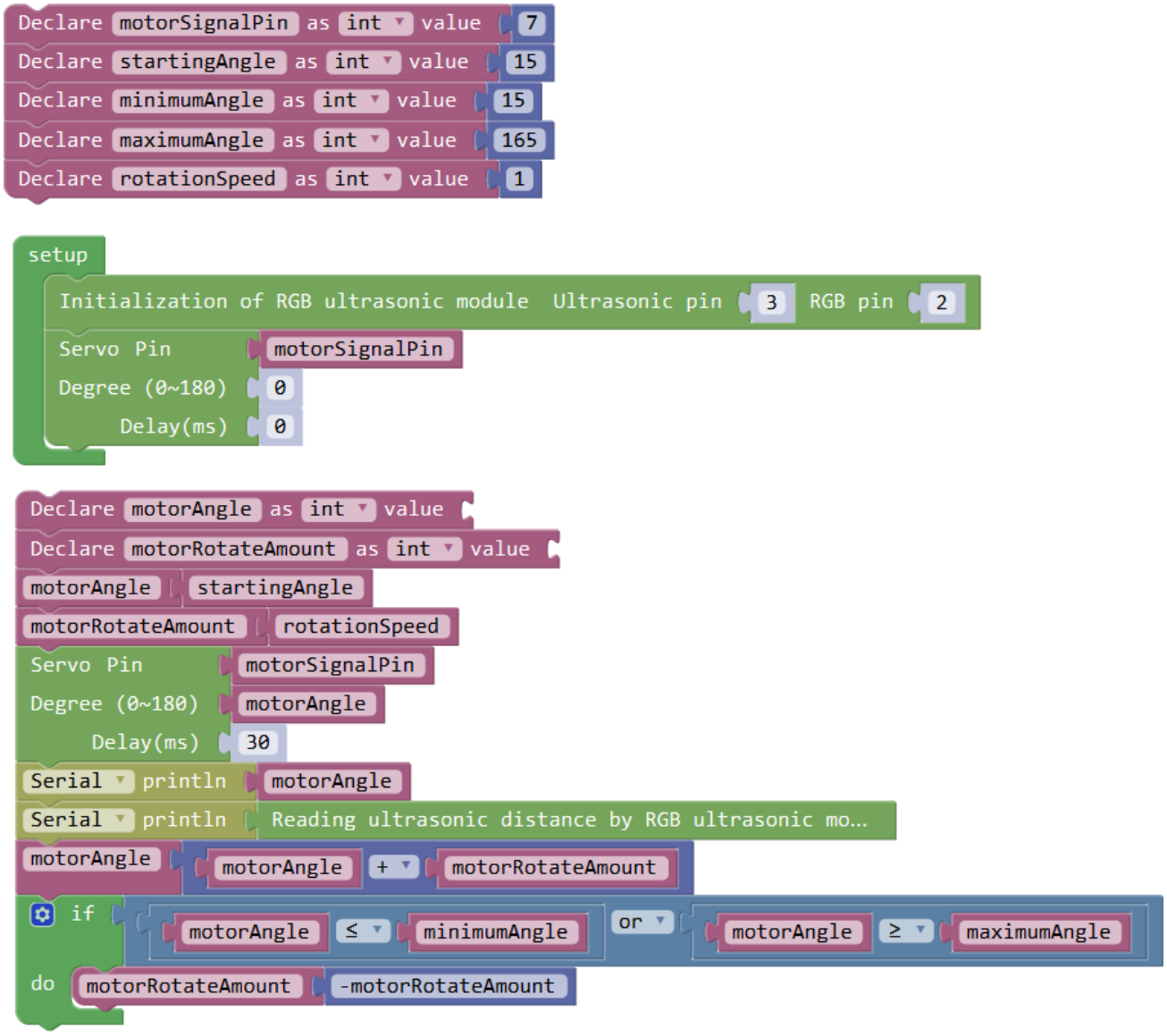

90°
120°
60°
150°
30°
10cm
20cm
30cm
40cm
Object: In Range
Angle: 94 °
Distance: 31 cm

Физический результат



Программа графического программирования Mixly

Mixly пишет программу ультразвукового радара RGB, как показано ниже:



Программа графического программирования MagicBlock

MagicBlock пишет программу ультразвукового радара RGB, как показано ниже: