基于 Arduino 单片机的超声波扫描测距云台

16307110037 孙炫东

摘要 在本文中，作者基于 Arduino，将摇杆手柄、两轴云台、超声波测距仪结合在一起。装置可以使用摇杆手动测量某一方向的距离，也可以自动扫描某一角度范围内的二维平面，并通过 python 语言根据数据大致重建平面的形状。根据实验结果，此套装置最佳实验范围为距离平面 20-50cm，测量时各个测量方向的入射角应小于 40°。

一、引言

Arduino 是一款可重复烧录程序的开源单片机。Arduino 拥有多个数字和模拟的输入输出端口，可同时控制多个探测器、舵机、LED 等元器件。在本实验中，作者基于 Arduino UNO 单片机在两轴云台上安装了一个超声波测距仪。可以使用摇杆手动控制云台转向，对特定方向进行测距。也可以通过程序控制，让云台自动完成一定角度范围内的扫描测距，并且利用 python 处理回传给电脑的测距数据，还原出扫描范围内大致的二维图像。

二、实验原理

超声波测距的原理如下。如图 1，超声波测距仪有一个发射器和一个紧挨着的接收器。一个测量周期内，发生器发出超声波脉冲（市面上的仪器常用脉冲频率为 40kHz），脉冲遇到反射物会反射回测距仪，被接收器接收。测量从脉冲发出到接收到回波的时间间隔 t，即可算出测距仪到反射物的距离 L=vt/2，其中 v 是测量环境中的声速。

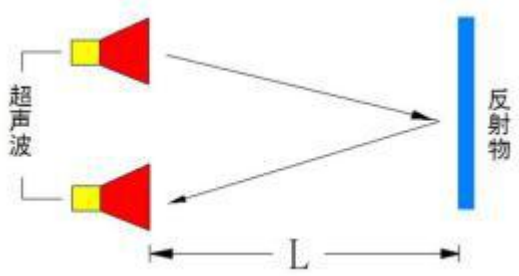


图 1

三、实验装置

实验装置的实物图和接线图如图 2 所示。

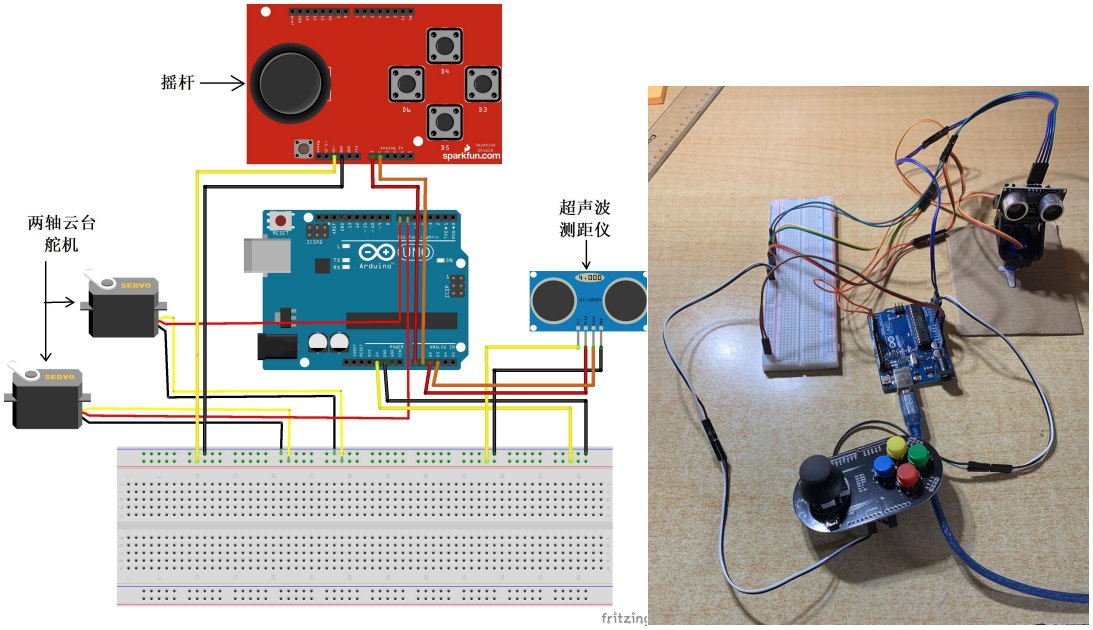


图 2

1.两轴云台

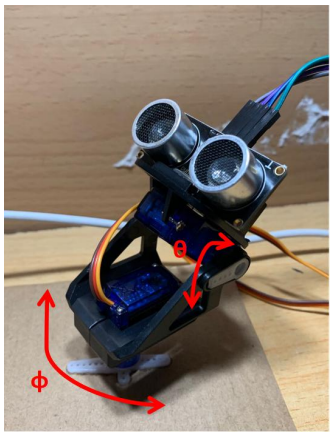


图 3

如图 3 所示，两轴云台由两个互相垂直放置的 5g 舵机组成。他们分别控制水平角度 ϕ 和竖直方向的角度 θ 。舵机的可变范围是 0-180°，由 Arduino 的两个数字输出端口提供目标角度，由此云台可以指向上半球的绝大部分方向。

2.摇杆[1]

图 4

如图 4 所示，摇杆在使用过程中会读取 XY 两个方向的偏转角度，发送角度到 Arduino 的两个模拟输入端口。Arduino 将输入电压划分为 1-1024 级。按照作者的设计，1-500 级对应角度减小，524-1024 级对应角度增大，其余部分（大约对应 4.2°）对应角度不变。

3.超声波测距仪[2]

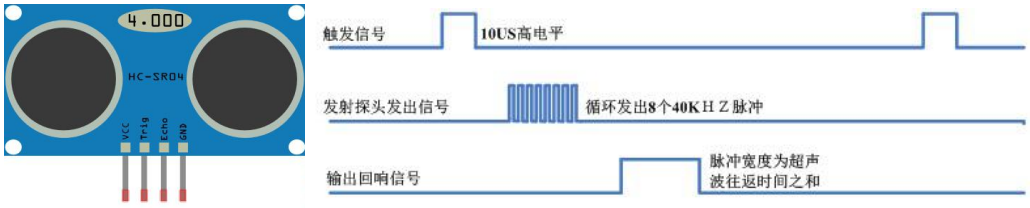


图 5

超声波测距仪的结构与工作时序如图 5 所示。首先 Arduino 通过一个数字输出端口向超声波测距仪 Trig 端口发送 10ms 的高电平，激发测距仪工作。接着发射探头发出八个 40kHz 的脉冲，并记录从发出到接收探头收到回声的时间间隔 t。探测完成后，通过 Echo 端口回传持续时间与 t 相同的高电平到 Arduino 的一个数字输入端口。

四、实验过程

1.手动摇杆测距

1）将测距仪放在竖直平面前，用直尺量出测距仪到平面的距离。

2）使用摇杆改变测距仪角度，直到测量显示的距离最小时记录读数，作为测得的到平面的距离。

3）不断改变测距仪到平面的距离，对比测得读数和实际距离的偏差。

2.自动扫描探测

1）将测距仪放在墙面前。

2）按照程序设定，云台按Δφ=3°，Δθ=3°的间隔，扫过45≦φ≦135°，0≦θ≦45°的范围。记录在每个角度超声波测距的结果。

3）利用测得的数据，在 python 中绘制三维重建图像，和实际墙面的形状做对比。

4）测量不同形状的墙面，重复以上实验。

五、实验结果与分析

1.手动摇杆测距

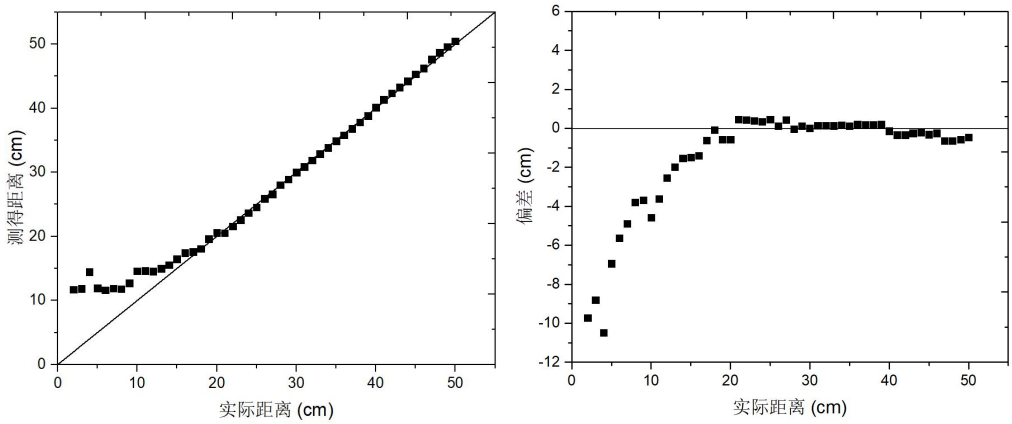
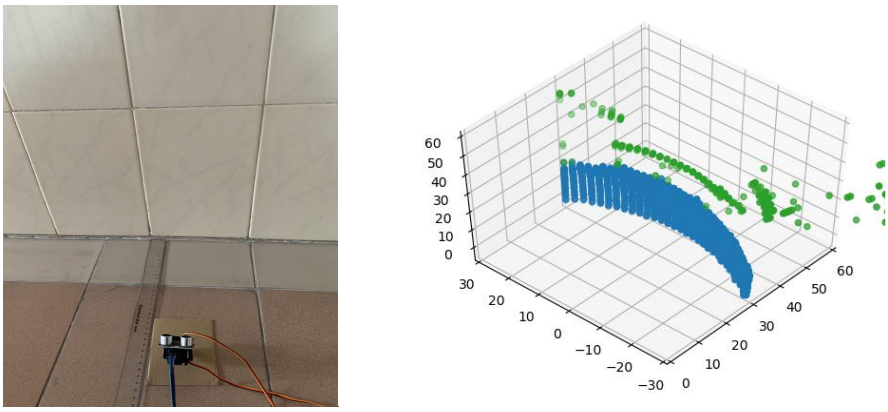


图 6

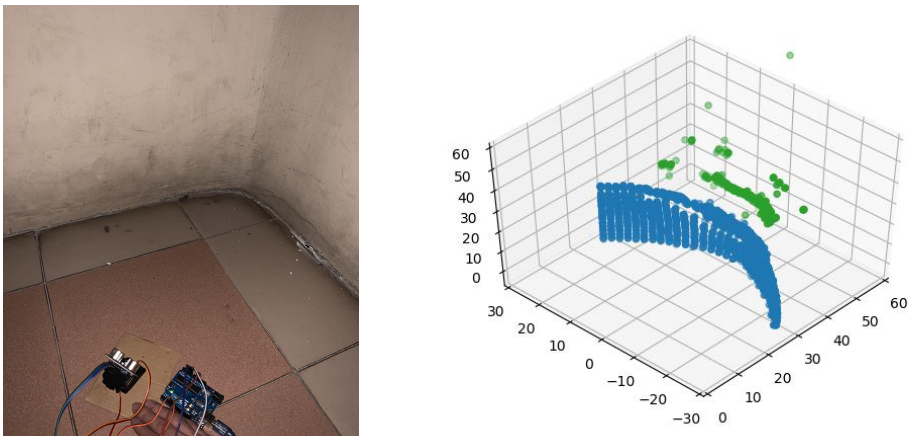
测距结果如图 6 所示。当超声波测距仪距离平面 0-20cm 时，测得的结果有较大误差。距离 20-50cm 时，测距结果较为准确，误差大约在±0.5cm 范围内。0-20cm 范围内误差较大的原因是，当 L=20cm 时，激发脉冲和接收回声的间隔时间为 t=2L/v≈1.18ms。所以当 L<20cm 时，t 可能不足 1ms。t 时长太短，导致 Arduino 测量高电平持续时间时产生了较大误差。

2.自动扫描探测

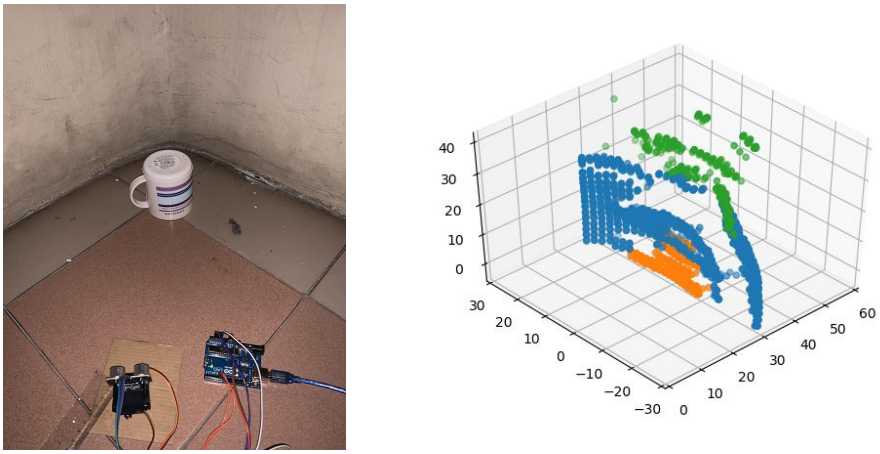
在自动扫描探测中，作者分别探测了平面、90°圆弧墙角、90°圆弧墙角+障碍物，三种不同的墙面，结果如图 7。



（a）平面



（b）转角

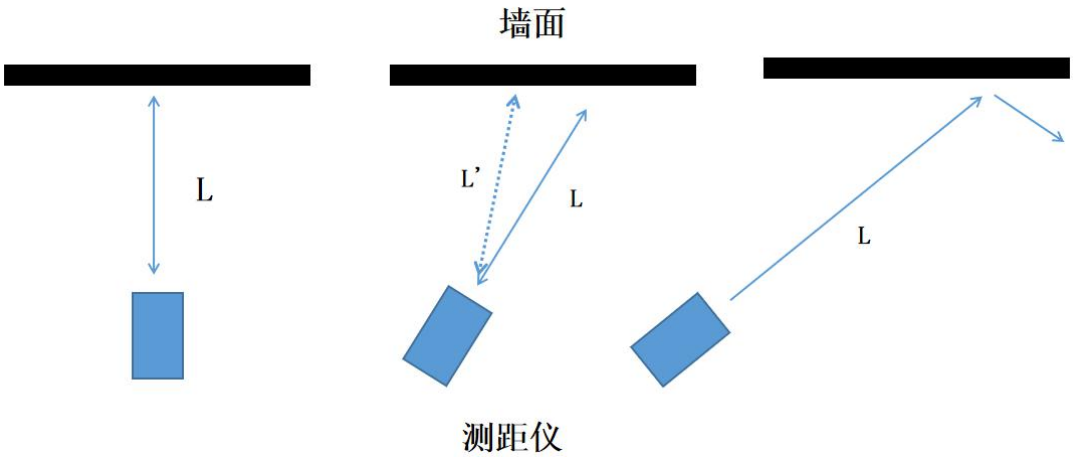


（c）转角+障碍物

图 7 橙色点 L<33cm；蓝色点 33<L<45cm；绿色点 L>45cm

在图 7（a）中可以看到，实际墙面是平面，但是扫描得到的平面略有弧度。这是因为，如图 8b，超声测距的定向性较差。超声脉冲会在发出后发散，当入射角大于 0 时，实际测到为虚线长度 L’，小于实际长度 L。这种偏差导致平面扫描结果为弧形。

还可以看到，图 7 中绿点中的部分数据测得的距离远大于实际距离 L。这是因为，如图 8c，当入射角过大时（实验中测得入射角大于越 40°），绝大部分回声无法返回至测距仪，因此测距仪接收不到回波，回声时间 t 过长，测得的 L 过大。



（a）　　（b）　　（c）

图 8

六、实验结论

1.超声波测距仪在垂直入射测量竖直平面时，最佳探测范围为 20-50cm。

2.超声波测距仪在扫描测距时，需要注意当入射角为 0-40° 时，由于超声波定向性较差，测得的距离会小于实际距离；当入射角>40° 时，由于超声波脉冲无法返回，测得的距离会远大于实际距离。

七、参考文献

【1】如何使用 Arduino 摇杆模块（Joystick Shield）https://www.yiboard.com/thread-913-1-1.html

【2】Arduino 中使用超声波测距实验 https://blog.csdn.net/haigear/article/details/84895381

附录

1.手动摇杆测距程序：

```
#include <Servo.h>
#define PIN_SERVO_down 6
#define PIN_SERVO_up 7
#define x_control A0
#define y_control A1
Servo myservo_down;
Servo myservo_up;
int x_motor=90;
int y_motor=30;
//echo setting
int TrgPin = A2;
int EcoPin = A3;
float dist;
int flag=1;


void setup() {

myservo_up.attach(PIN_SERVO_up);

myservo_down.attach(PIN_SERVO_do
wn);
  int x_motor=90;
  int y_motor=30;

  Serial.begin(9600);
//设置 TrgPin 为输出状态
pinMode(TrgPin, OUTPUT);
// 设置 EcoPin 为输入状态
pinMode(EcoPin, INPUT);
}

void loop() {
  flag=1;
  int x_read = analogRead(x_control);
  int y_read = analogRead(y_control);
  if (x_read<500)
{x_motor-=1;flag=0;}
  if (x_read>524)
{x_motor+=1;flag=0;}
  if (y_read<500)
{y_motor-=1;flag=0;}
  if (y_read>524)
{y_motor+=1;flag=0;}
  if (x_motor<0)  {x_motor=0;}
  if (x_motor>180)  {x_motor=180;}
  if (y_motor<0)  {y_motor=0;}
  if (y_motor>100)  {y_motor=100;}
  myservo_down.write(x_motor);
  myservo_up.write(y_motor);
  delay(10);
if (flag==1){
  digitalWrite(TrgPin, LOW);
  delayMicroseconds(8);
  digitalWrite(TrgPin, HIGH);
  // 维持 10 毫秒高电平用来产生一
个脉冲
  delayMicroseconds(10);
  digitalWrite(TrgPin, LOW);
  // 读取脉冲的宽度并换算成距离
  dist = pulseIn(EcoPin, HIGH) / 58.30;
  Serial.print("Distance:");
  Serial.print(dist);
  Serial.println("cm");
  delay(200);
  }

}
```

2.自动扫描探测程序：

```
#include <Servo.h>
#define PIN_SERVO_down 6
#define PIN_SERVO_up 7
#define x_control A0
#define y_control A1
Servo myservo_down;
Servo myservo_up;
int x_motor=90;
int y_motor=30;
//echo setting
int TrgPin = A2;
int EcoPin = A3;
float dist,sum;
int flag=1;
int x,y,i;
//每一点的采样次数
int N=1;


void setup() {

myservo_up.attach(PIN_SERVO_up);

myservo_down.attach(PIN_SERVO_do
wn);
  int x_motor=90;
  int y_motor=30;

  Serial.begin(9600);
//设置 TrgPin 为输出状态
pinMode(TrgPin, OUTPUT);
// 设置 EcoPin 为输入状态
pinMode(EcoPin, INPUT);
}

void loop() {
  delay(2000);
  Serial.println("x  y
distance(cm)");
  flag=1;
  x=45;
  y=0;
  while(y<=45){
    x=45;
    while(x<=135){
      x_motor=x;
      y_motor=y+2;

      if (x_motor<0)  {x_motor=0;}
      if  (x_motor>180)
{x_motor=180;}
      if (y_motor<0)  {y_motor=0;}
      if  (y_motor>100)
{y_motor=100;}


myservo_down.write(x_motor);
      myservo_up.write(y_motor);
      if(x==45){delay(800);}
      delay(200);
```

```
        sum=0;
        for (i=1;i<=N;i++){
          digitalWrite(TrgPin, LOW);
          delayMicroseconds(8);
          digitalWrite(TrgPin, HIGH);
          // 维持 10 毫秒高电平用来
产生一个脉冲
          delayMicroseconds(10);
          digitalWrite(TrgPin, LOW);
          // 读取脉冲的宽度并换算
成距离
          dist = pulseIn(EcoPin, HIGH)
/ 58.30;
          sum+=dist;
          Serial.print(x);
          Serial.print(" ");
          Serial.print(y);
          Serial.print(" ");
          Serial.println(dist);
          delay(50);
          }
        dist=sum/N;
        Serial.print(x);
        Serial.print(" ");
        Serial.print(y);
        Serial.print(" ");
        Serial.println(dist);
        delay(50);

        x+=3;
        }

    y+=3;
    }
   delay(60000);
}
```

3.平面重建程序（python 语言）：

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import
Axes3D

fig = plt.figure()
ax1 = plt.axes(projection='3d')
plt.xlim(0, 60)
plt.ylim(-30, 30)

fx=4#y 角度修正

#读文件
f=open("5-1.txt")
data=[]
for t in f.readlines():
    t1=t.split( )
    t2=[]
    for i in t1:
        t2.append(float(i))
    data.append(t2)
f.close()



import numpy as np
xd=[]
yd=[]
zd=[]
xd1=[]
yd1=[]
zd1=[]
xd2=[]
yd2=[]
zd2=[]
for i in data:
    if i[2]<100:

z=i[2]*np.sin((i[1]-fx)/180*np.pi)

x=i[2]*np.cos((i[1]-fx)/180*np.pi)*np.c
os((i[0]-90)/180*np.pi)

y=i[2]*np.cos((i[1]-fx)/180*np.pi)*np.s
in((i[0]-90)/180*np.pi)
        #if (x<27)and(z<20):
        if (i[2]>33)and(i[2]<45):
            xd.append(x)
            yd.append(y)
            zd.append(z)
        if (i[2]<33):
            xd1.append(x)
            yd1.append(y)
            zd1.append(z)
        if (i[2]>45):
            xd2.append(x)
            yd2.append(y)
            zd2.append(z)


ax1.scatter3D(xd,yd,zd,  cmap='Blues')
#绘制散点图
ax1.scatter3D(xd1,yd1,zd1,
cmap='Red')  #绘制散点图
ax1.scatter3D(xd2,yd2,zd2,
cmap='Yellow')  #绘制散点图


plt.show()
```