基于 arduino 的超声波声纳制作

17307110121 陈巍文

一、课题设计

超声波传感器和舵机是 arduino 常见的外设，本课题使用这两个外设，搭建了简单有效的声纳系统，可以自动探测周围的环境。

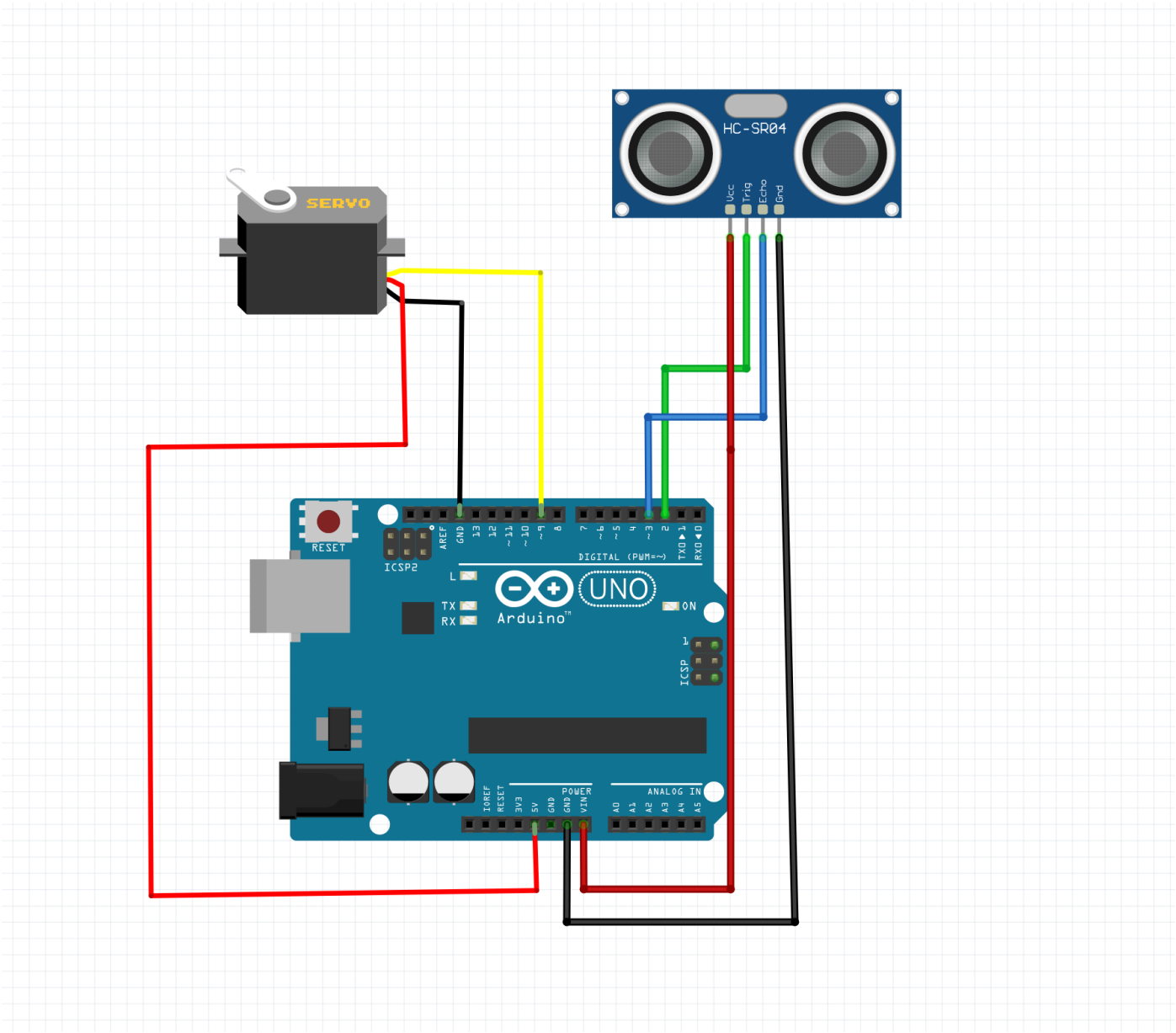
参考了 Arduino Sonar © MIT - Arduino Project Hub，并进行了改进，增加了一个可以跨平台运行的监视器程序。

二、实验装置及过程

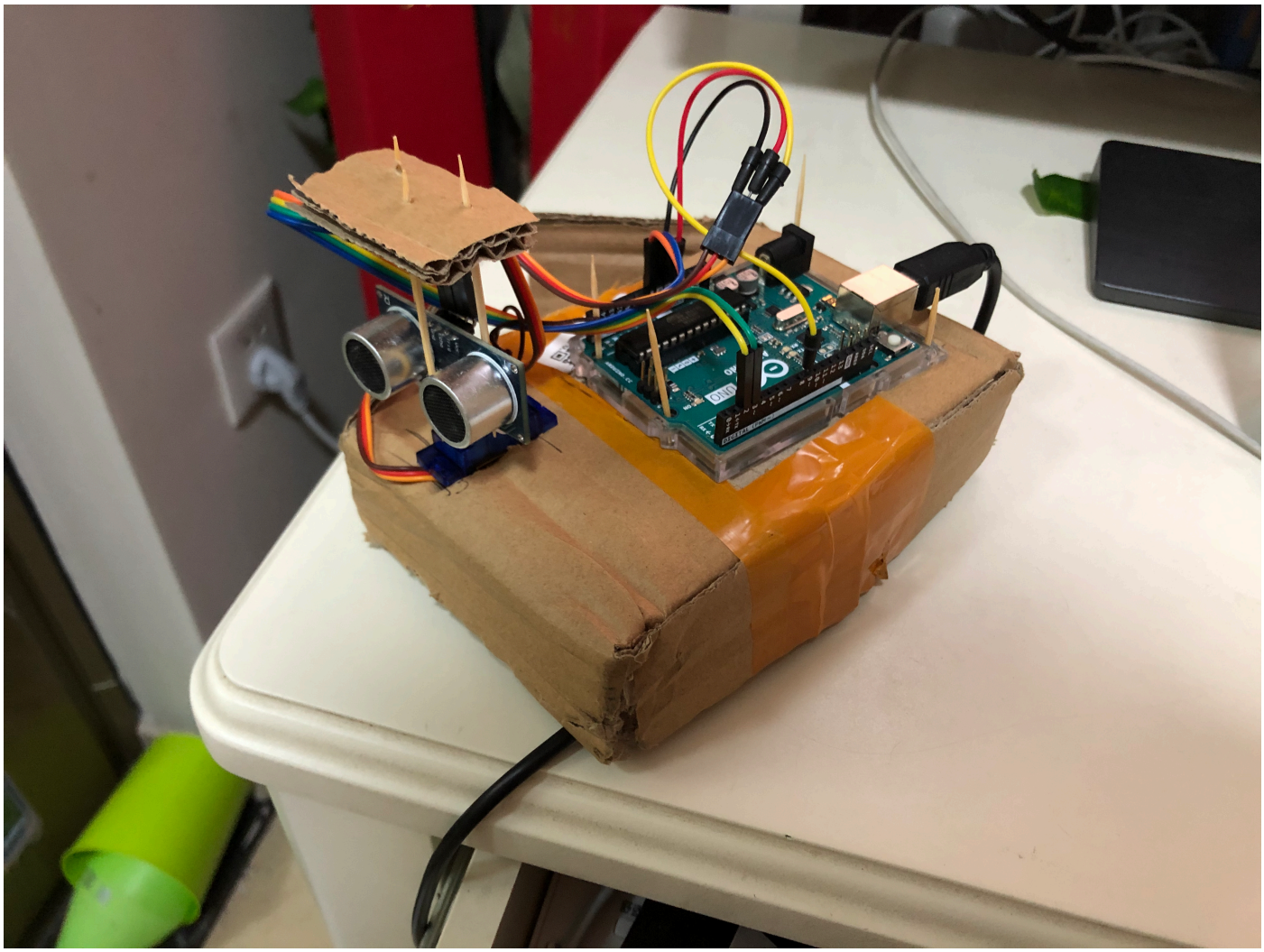
元件：

- Arduino Uno
- Ultrasonic Sensor HC-SR04
- Servo TS90A

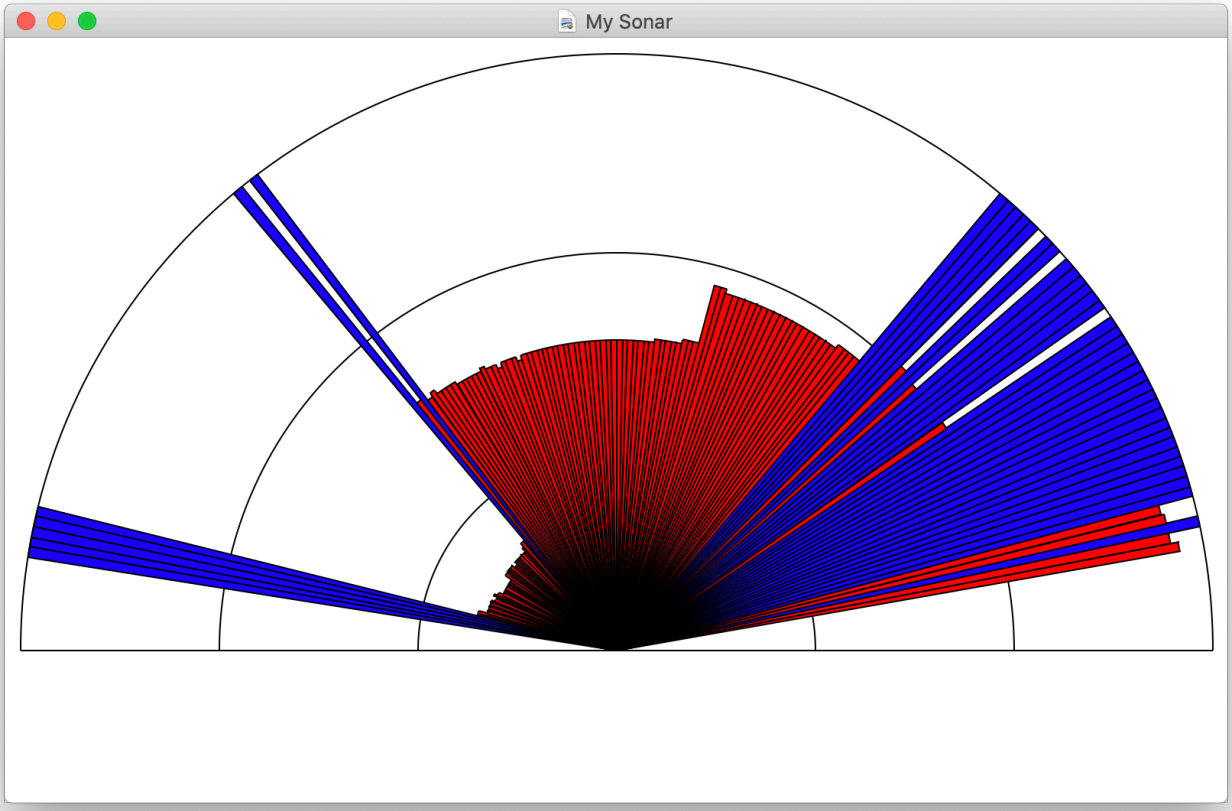
连线图：


HC-SR04
Vcc
Trig
Echo
Gnd
SERVO
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

实拍图：



监视器使用 python 开发：



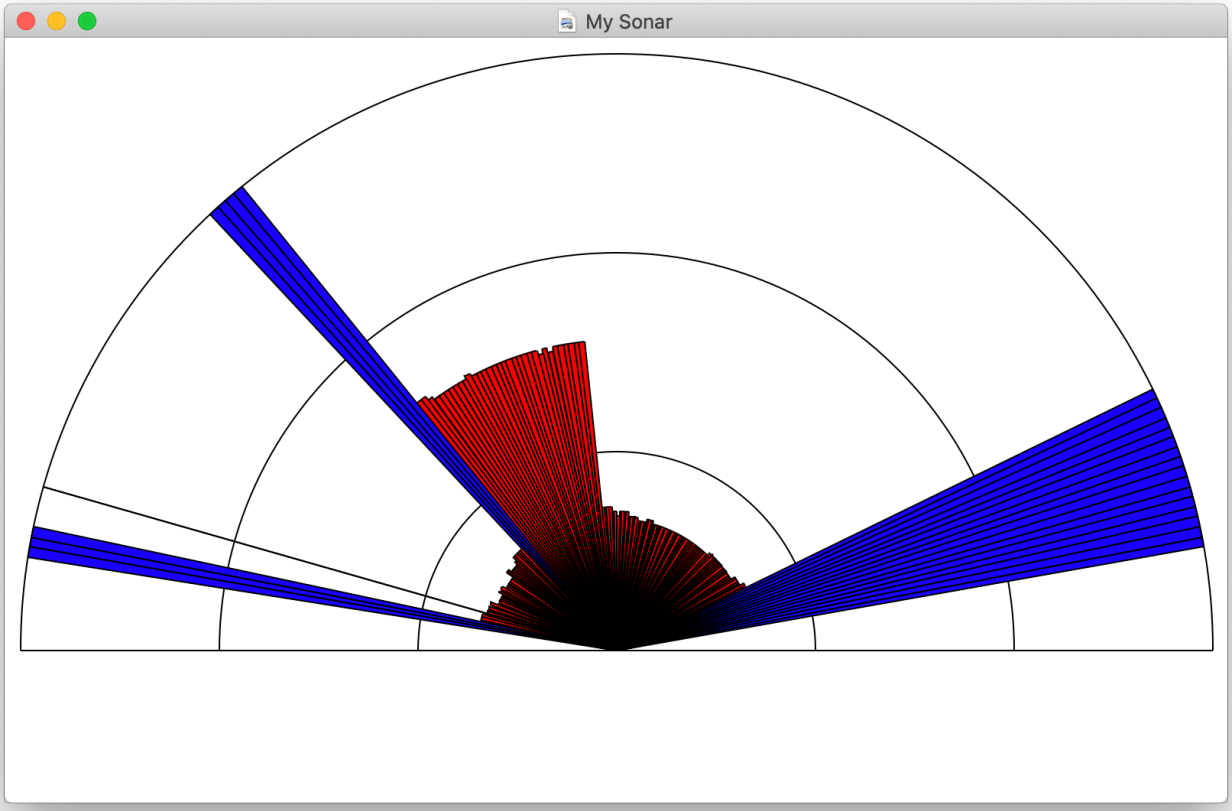
三环分别表示 50，100，150 cm。

三、实验结果

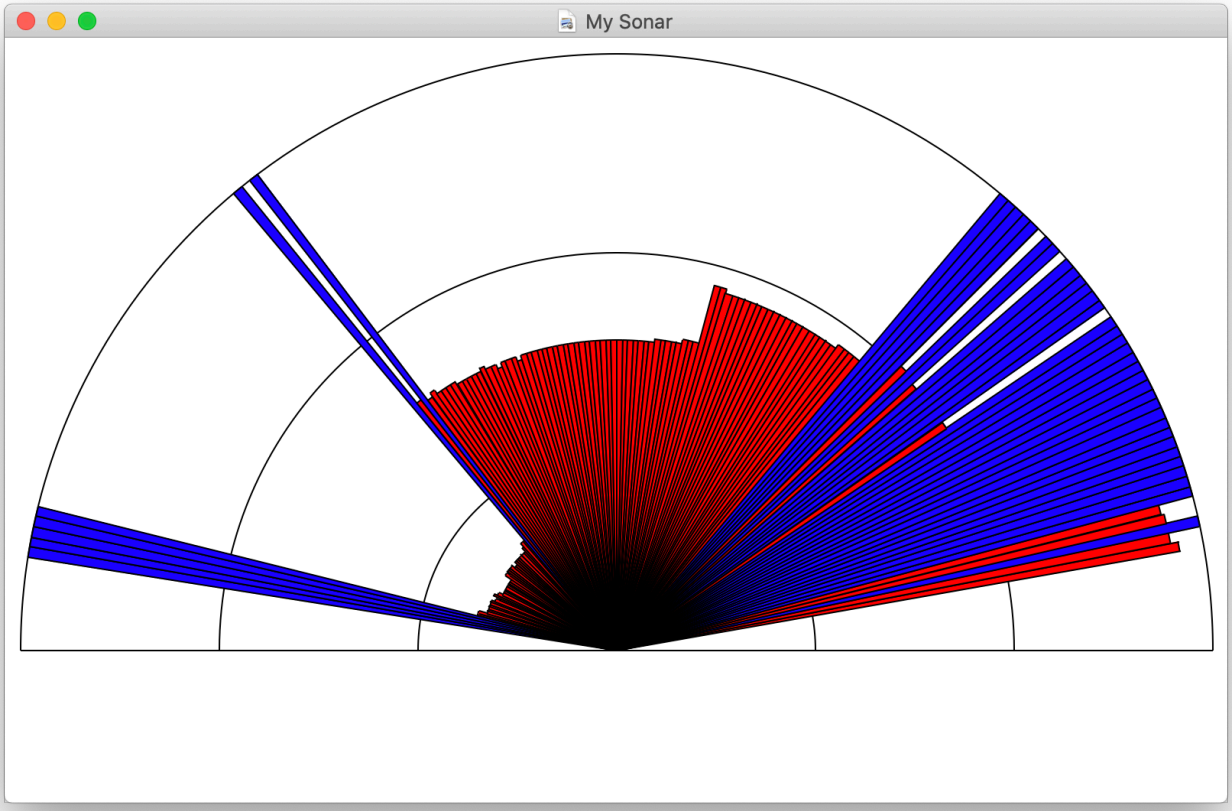
能够大致显示出障碍物的图像，在保证读数质量的情况下，一次扫描的时间约为 5 秒。

以下是两种障碍物放置结构，用于后续的分析


西洋参补
纯正天然
LANSHENG


My Sonar


西洋参补
純正天然
GIANGSHENG



图像能观察到两种明显的畸变：

1. 平面障碍物被显示为弧面；
2. 障碍物的角宽度比显示的大。

通过分析，这两种畸变来自同一个原因：发射超声波有一个角度，同时距离读数只取最先返回的回波。

四、实验分析

发射超声波并不是完美的指向性，而是有一个小的角度θ；同时距离读数是按照最先返回的声波来计算。

障碍物的角宽度比原来大，原因就是在发射超声波的小角度覆盖到障碍物时就开始显示读数，所以障碍物的测定角宽度会比真实角宽度多两个θ；

平面障碍物显示为弧面，尤其是角宽度小的障碍物非常明显，也是因为发射的超声波有角度，而距离读数取这个角度中最近的值。

附录：Code

Arduino 代码，引用了[Arduino Sonar © MIT - Arduino Project Hub]。

```
/*
https://www.hackster.io/faweiz/arduino-radar
```

```
Radar Screen Visualisation for HC-SR04
Sends sensor readings for every degree moved by the servo
values sent to serial port to be picked up by Processing
*/

#include <NewPing.h>
#include <Servo.h>

#define TRIGGER_PIN  2   // Arduino pin 2 tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     3   // Arduino pin 3 tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 150 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.
#define SERVO_PWM_PIN 9 //set servo to Arduino's pin 9

// means -angle .. angle
#define ANGLE_BOUNDS 80
#define ANGLE_STEP 1

int angle = 0;

// direction of servo movement
// -1 = back, 1 = forward
int dir = 1;

Servo myservo;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
  Serial.begin(9600); // initialize the serial port:
  myservo.attach(SERVO_PWM_PIN); //set servo to Arduino's pin 9
}

void loop() {

  delay(50);
  // we must renormalize to positive values, because angle is from -ANGLE_BOUNDS .. ANGLE_BOUNDS
  // and servo value must be positive
  myservo.write(angle + ANGLE_BOUNDS);

  // read distance from sensor and send to serial
  getDistanceAndSend2Serial(angle);

  // calculate angle
  if (angle >= ANGLE_BOUNDS || angle <= -ANGLE_BOUNDS) {
    dir = -dir;
  }
  angle += (dir * ANGLE_STEP);
}

int getDistanceAndSend2Serial(int angle) {
```

```
  int cm = sonar.ping_cm();
  Serial.print(angle, DEC);
  Serial.print(",");
  Serial.println(cm, DEC);

}
```

监视器代码:

```
'''
@Author: Ofey Chan
@Date: 2020-05-11 17:09:43
@LastEditors: Ofey Chan
@LastEditTime: 2020-05-11 20:06:33
@Description: gui.py
@Reference:
'''

from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from constants import *
from serial_utils import find_serial_port, get_platform_name
import serial



# Convert coordinate of sphere from (x_center, y_center, radius) to (x0, y0, x1, y1) form
def center_to_xxyy(x_center: int, y_center: int, radius: int) -> (int, int , int, int):
    return (
        x_center - radius,
        y_center - radius,
        x_center + radius,
        y_center + radius
    )


# A very c-flavor getdata.
def getdata(data: list) -> bool:
    tokens = serial.readline().decode('utf-8').split(',')

    if len(tokens) == 2:
        data.append(int(tokens[0]))
        data.append(int(tokens[1].rstrip()))
        return True
    else:
        return False


all_fans = {}
```

```
indicator = None

def draw_radar_fan(radar_map: Canvas, angle: int, distance: int) -> None:
    global indicator
    if str(angle) in all_fans:
        radar_map.delete(all_fans[str(angle)])
    if indicator:
        radar_map.delete(indicator)

    fan_coordinate = center_to_xxyy(
        x_center=RADAR_CHART_CENTER_X,
        y_center=RADAR_CHART_CENTER_Y,
        radius=RADAR_CHART_RADIUS * distance / MAX_DETECT_DISTANCE
    )

    indicator_coordinate = center_to_xxyy(
        x_center=RADAR_CHART_CENTER_X,
        y_center=RADAR_CHART_CENTER_Y,
        radius=RADAR_CHART_RADIUS
    )

    if distance == 0:
        fan = radar_map.create_arc(*indicator_coordinate, start=angle+90, extent=1,
fill='blue', tags=str(angle))
    else:
        fan = radar_map.create_arc(*fan_coordinate, start=angle+90, extent=1, fill='red',
tags=str(angle))

    indicator = radar_map.create_arc(*indicator_coordinate, start=angle+90, extent=0,
tags='indicator')
    all_fans[str(angle)] =fan


def getdata_redraw_loop(root: Tk, radar_map: Canvas) -> None:
    data = []
    if getdata(data):
        angle, distance = data
        print(angle, ' ', distance)

        draw_radar_fan(radar_map, angle, distance)
        radar_map.pack()

    root.after(50, getdata_redraw_loop, root, radar_map)



def create_radar_map(root: Tk) -> Canvas:
    canvas = Canvas(master=root, width=CANVAS_WIDTH, height=CANVAS_HEIGHT)
    ring1_coordinate = center_to_xxyy(
        x_center=RADAR_CHART_CENTER_X,
        y_center=RADAR_CHART_CENTER_Y,
```

```
        radius=RADAR_CHART_RADIUS
    )
    ring2_coordinate = center_to_xxyy(
        x_center=RADAR_CHART_CENTER_X,
        y_center=RADAR_CHART_CENTER_Y,
        radius=RADAR_CHART_RADIUS * 2 / 3
    )
    ring3_coordinate = center_to_xxyy(
        x_center=RADAR_CHART_CENTER_X,
        y_center=RADAR_CHART_CENTER_Y,
        radius=RADAR_CHART_RADIUS / 3
    )

    canvas.create_arc(*ring1_coordinate, start=0, extent=180, tags='ring1')
    canvas.create_arc(*ring2_coordinate, start=0, extent=180, tags='ring2')
    canvas.create_arc(*ring3_coordinate, start=0, extent=180, tags='ring3')

    return canvas


root = Tk()

root.geometry(str(DEFAULT_WINDOW_WIDTH) + 'x' + str(DEFAULT_WINDOW_HEIGHT))
root.title(WINDOW_TITLE_STRING)
root.iconbitmap('icons/pypad.ico')

radar_map = create_radar_map(root)

radar_map.pack()

port = find_serial_port(get_platform_name())[0]

serial = serial.Serial(port, 9600, timeout=0.06)

root.after(50, getdata_redraw_loop, root, radar_map)
root.mainloop()

# Finish
print("Finish!")
serial.close()

```

```
'''
@Author: Ofey Chan
@Date: 2020-05-11 09:33:53
@LastEditors: Ofey Chan
@LastEditTime: 2020-05-11 10:33:47
@Description: serial_utils.py
@Reference:
'''

```

```
import pathlib
import serial
import platform
from glob import glob
from time import sleep

# Return OS type.
# eg: 'Linux', 'Darwin', 'Java', 'Windows'
def get_platform_name():
    return platform.system()

darwin_serial_patterns = [
    '/dev/tty.usbmodem*',
    '/dev/tty.usbserial*'
]

linux_serial_patterns = [
    ' /dev/ttyUSB*',
    '/dev/ttyACM*'
]

# Return absolute dir of all possible serial ports.
def find_serial_port(platform_name='Darwin'):
    if platform_name == 'Darwin':
        possible_port = []
        for pattern in darwin_serial_patterns:
            possible_port.extend(glob(pattern))
        return possible_port
    elif platform_name == 'Linux':
        possible_port = []
        for pattern in linux_serial_patterns:
            possible_port.extend(glob(pattern))
        return possible_port
    # TODO: Add Windows® support.
    else:
        raise Exception( 'Platform \'{}\' is not supported by this function, please find serial port by yourself.'.format(platform_name) )


# For test.
def print_output():
    serial_port = find_serial_port(platform_name=get_platform_name())[0]
    print(serial_port)
    with serial.Serial(serial_port, 9600, timeout=1) as ser:
        while True:
            line = ser.readline()
            if line:
                print(line)
            sleep(0.05)

```

```
def main():
    print_output()


if __name__ == '__main__':
    main()

```

```
'''
@Author: Ofey Chan
@Date: 2020-05-11 13:50:02
@LastEditors: Ofey Chan
@LastEditTime: 2020-05-11 20:09:20
@Description: constants.py
@Reference:
'''

DEFAULT_WINDOW_WIDTH = 800
DEFAULT_WINDOW_HEIGHT = 500

CANVAS_WIDTH = DEFAULT_WINDOW_WIDTH
CANVAS_HEIGHT = DEFAULT_WINDOW_HEIGHT

RADAR_CHART_BORDER_WIDTH = 10
RADAR_CHART_RADIUS = CANVAS_WIDTH / 2 - RADAR_CHART_BORDER_WIDTH
RADAR_CHART_CENTER_X = RADAR_CHART_BORDER_WIDTH + RADAR_CHART_RADIUS
RADAR_CHART_CENTER_Y = RADAR_CHART_BORDER_WIDTH + RADAR_CHART_RADIUS

DEFAULT_DATA_LIMIT = 10000

# Parameters about Sonars
MAX_DETECT_DISTANCE = 150  # in cm.


# Strings
WINDOW_TITLE_STRING = 'My Sonar'
```