```
# Script for running the LIDAR program.
# Written by Jack Fan and David Zhu.

# Load packages.
from time import sleep
import serial
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from math import *

# Objects for simplifying our lives.
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

class Data(object):
    def __init__(self, x, y, value, distance):
        self.x = x
        self.y = y
        self.value = value
        self.distance = distance

class coordinate(object):
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

# Function for correcting input bytes to calibrated distance.
def finddistance(n):
    n = int(n)
    result = y = 11194*n**-1.151

    return result

# Set definitions.
ser = serial.Serial(port = "/dev/ttyACM1", baudrate = 9600, timeout = 0.1)
base_servo_start_pos = 0
base_servo_end_pos = 180
base_servo_pos = 0
sensor_servo_start_pos = 0
sensor_servo_end_pos = 180
sensor_servo_pos = 0
data1 = []
data2 = []
value = 0
distance = 0
ser.close()
ser.open()
xs = []
ys = []
zs = []
fx = open('xlist.txt','w')    # Create text documents for backup storage of information.
fy = open('ylist.txt','w')
fz = open('zlist.txt','w')

# Input angle ranges for scan.
base_servo_start_pos = int(input("base_servo_start_pos="))
base_servo_end_pos = int(input("base_servo_end_pos="))
```

```
sensor_servo_start_pos = int(input("sensor_servo_start_pos="))
sensor_servo_end_pos = int(input("sensor_servo_end_pos="))

# Scan and collect data.
for i in range(base_servo_start_pos, base_servo_end_pos + 1):
    if i>=10 and i<=99:
        base_servo_pos = "0" + str(i)
    elif i<=9:
        base_servo_pos = "00" + str(i)
    else: base_servo_pos = str(i)
    for j in range(sensor_servo_start_pos, sensor_servo_end_pos + 1):
        if j>=10 and j<=99:
            sensor_servo_pos = "0" + str(j)
        elif j<=9:
            sensor_servo_pos = "00" + str(j)
        else: sensor_servo_pos = str(j)
        if ser.writable:
            ser.write(base_servo_pos+"a")
        sleep(0.001)
        if ser.writable:
            ser.write(sensor_servo_pos+"a")
        sleep(0.001     )
        if ser.readable:
            value = ser.read(7)
        value = value[0:-2]
        distance = finddistance(value)
        distance = round(distance,2)
        if distance>=8 and distance <=60:
            data1.append(Data(float(base_servo_pos),float(sensor_servo_pos),float
(value),distance))
        print(i,j,base_servo_pos,sensor_servo_pos,value,distance)

# Convert spherical to cartesian.
for i in data1:
    data2.append(coordinate(i.distance*cos(i.y/180.0*pi)*cos(i.x/180.0*pi),i.distance*cos
(i.y/180.0*pi)*sin(i.x/180.0*pi),i.distance*sin(i.y/180.0*pi)))

# Store coordinates into backup text files.
for i in range(len(data2)):
    xs.append(data2[i].x)
    fx.writelines(str(data2[i].x)+'\n')
    ys.append(data2[i].y)
    fy.writelines(str(data2[i].y)+'\n')
    zs.append(data2[i].z)
    fz.writelines(str(data2[i].z)+'\n')

# Close our storage files.
fx.close()
fy.close()
fz.close()

# Plot.
ax.scatter(xs,ys,zs,s=5,c='r',marker='o')  # Creates a scatter plot.
#ax.plot_wireframe(xs,ys,zs) # Creates a wireframe plot.
# ax.plot_surface(xs,ys,zs)     # Creates a surface plot.
plt.xlim([0,20])      # Set explicit boundaries on the plot.
plt.ylim([0,20])
ax.set_zbound(0,30)
plt.show() # Plot.
```