



Universidad Nacional Autónoma de México

Colegio de Ciencias y Humanidades

Profesora: Plata Luna Iveth Vanessa

Proyecto: Radar con Arduino y Python

Fecha: 30/07/23

Correo: iveth.plata@cch.unam.mx

Plantel: CCH Oriente

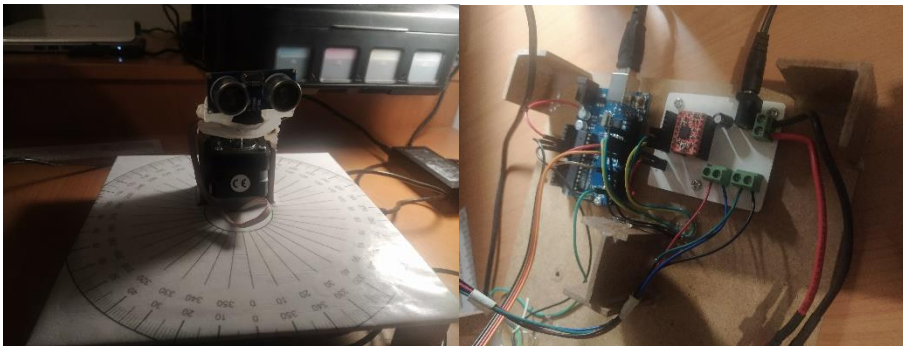
Diplomado en programación

Radar con Arduino y Python

El presente proyecto es un radar, rastrea objetos a través de un sensor ultrasónico en un rango de 360 grados y 20 centímetros. Los datos obtenidos se envían por medio del puerto serial y son codificados y almacenados en listas para que posteriormente la computadora represente de manera gráfica y en formato texto la ubicación en coordenadas polares y cartesianas los objetos con respecto al sensor ultrasónico. La computadora también calculará las distancias entre los objetos rastreados.

El alcance de este proyecto no involucra el diseño del radar y la programación en Arduino. Solamente se ven aspectos de la programación en Python. Ver imagen 1 Radar ultrasónico (sensor ultrasónico, motor a pasos, placa Arduino).

Ver imagen 1 Radar ultrasónico (sensor ultrasónico, motor a pasos, placa Arduino).



A continuación, se presentan algunas imágenes y datos que procesa por medio del lenguaje de programación Python.

La imagen 2 representa las cadenas de carácter que recibe el programa Python desde el puerto serial.

Imagen2

```
['R1rD11dG8g\r\n', 'R2rD20dG64g\r\n', 'R3rD15dG108g\r\n', 'R4rD17dG157g\r\n', 'R5rD6dG275g\r\n']
```

```

In [96]: ##Autor Plata Luna Iveth Vanessa
##Importamos la Librería para el uso del puerto serial y time par el manejo del tiempo.
import serial
import time

In [97]: ##El programa se conecta al puerto serial COM6 a 9600 baudios dando tiempos de demora para recibir datos.
serialArduino= serial.Serial("COM6", 9600 )
time.sleep(1)

In [98]: ##La variable cad almacenará de manera temporal Los datos obtenidos del puerto serial.
cad = ""

In [99]: ##Definimos una Lista para almacenar Los datos del puerto serial. La variable i es de control.
lista=[]
i = 0

In [102]: ##Segmento de código que obtiene datos desde el puerto serial y Lo transforma a ASCII. El bucle termina hasta que se obtiene una 'f'
while not(cad[0:1] == 'f') :
    cad = serialArduino.readline().decode('ascii')
    print(cad)
    if not(cad[0:1] == 'f'):
        lista.append(cad)
        i=i+1

```

Posteriormente el programa decodifica por medio de funciones para el manejo de cadenas de caracteres (búsqueda de caracteres y extracción de subcadenas). Al finalizar la decodificación se almacena los grados y la distancia de los objetos en listas. Ver imagen 3.

Imagen 3

```

R1rD11dG8g
R2rD20dG64g
R3rD15dG108g
R4rD17dG157g
R5rD6dG275g

índice de caracteres
[0, 0, 0, 0, 0]
[2, 2, 2, 2, 2]
[3, 3, 3, 3, 3]
[6, 6, 6, 6, 5]
[7, 7, 7, 7, 6]
[9, 10, 11, 11, 10]
Número de dígitos
[1, 1, 1, 1, 1]
[2, 2, 2, 2, 1]
[1, 2, 3, 3, 3]
Resultados
['1', '2', '3', '4', '5']
['11', '20', '15', '17', '6']
['8', '64', '108', '157', '275']

```

```

In [107]: #Definimos Listas que servirán a decodificar la información del puerto serial.
indexO = []
indexo = []
indexD = []
indexd = []
indexG = []
indexg = []
numDigO = []
numDigD = []
numDigG = []
objetoS = []
distanciaS = []
gradosS = []

#Segmento de código que decodifica las cadenas de caracter obtenidas desde el puerto serial.
#Se utiliza funciones para la manipulación de cadenas, como buscar un caracter y extraer subcadenas.
#Los resultados obtenidos se almacenan en listas.
for n in range(0, len(lista), 1):
    print(lista[n])
    cadena=lista[n]

    indexO.append(cadena.find('R'))
    indexo.append(cadena.find('r'))
    indexD.append(cadena.find('D'))
    indexd.append(cadena.find('d'))
    indexG.append(cadena.find('G'))
    indexg.append(cadena.find('g'))

    numDigO.append(indexo[n] - indexO[n] - 1)
    numDigD.append(indexd[n] - indexD[n] - 1)
    numDigG.append(indexg[n] - indexG[n] - 1)

    objetoS.append(cadena[(indexO[n]+1):(indexO[n]+1+numDigO[n])])
    distanciaS.append(cadena[(indexD[n]+1):(indexD[n]+1+numDigD[n])])
    gradosS.append(cadena[(indexG[n]+1):(indexG[n]+1+numDigG[n])])

print("índice de caracteres")
print(indexO)
print(indexo)
print(indexD)
print(indexd)
print(indexG)
print(indexg)
print("Número de dígitos")
print(numDigO)
print(numDigD)
print(numDigG)
print("Resultados")
print(objetoS)
print(distanciaS)
print(gradosS)

```

Continuando con el programa, con los datos obtenidos y almacenados (grados, distancia y objetos) se calcula las coordenadas cartesianas, para que posteriormente se represente un plano de manera gráfica. Ver imagen 4.

Imagen 4

```

[1, 2, 3, 4, 5]
[11, 20, 15, 17, 6]
[8, 64, 108, 157, 275]
[10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473]
[1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474]

```

```
In [108]: #En este segmento de código se convierten las coordenadas polares a coordenadas cartesianas.
#Para esta función utilizamos la librería math.
import math
objeto = []
distancia = []
grados = []

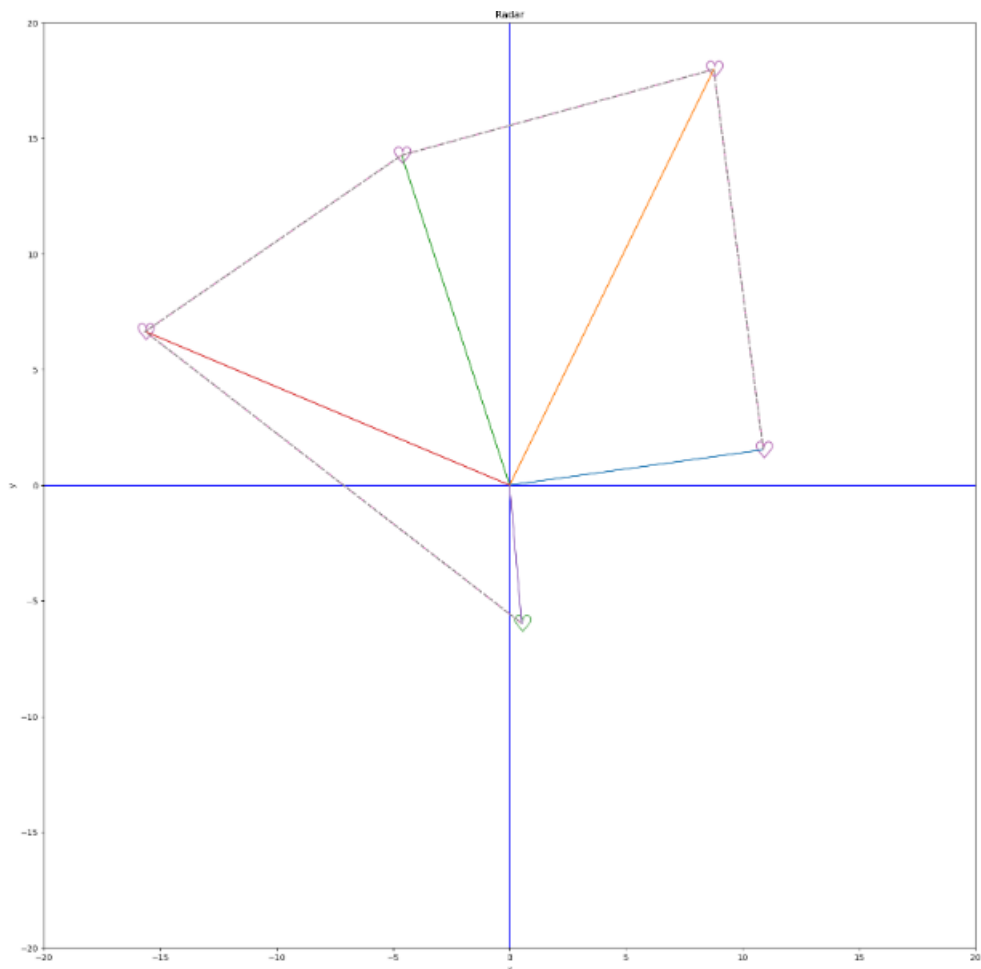
for n in range(0, len(lista), 1):

    objeto.append(int(objetos[n]))
    distancia.append(int(distancias[n]))
    grados.append(int(gradosS[n]))

x=[]
y=[]
for n in range(0, len(lista), 1):
    x.append(distancia[n] * math.cos(math.radians(grados[n])));
    y.append(distancia[n] * math.sin(math.radians(grados[n])));
print(objeto)
print(distancia)
print(grados)
print(x)
print(y)
```

Para graficar utilizamos la librería numpy y matplotlib. Utilizamos las listas de los objetos, grados, distancia y coordenadas para graficar. Ver imagen 5.

Imagen 5



```

In [109]: #Para graficar utilizamos las librerias mumpy y matplotlib.
import numpy as np
import matplotlib.pyplot as plt

In [110]: #Es importante convertir la lista a un tipo arreglo para poder graficar.
xnp = np.array(x)
ynp = np.array(y)
print(xnp)
print(ynp)

[ 10.89294876   8.76742294  -4.63525492 -15.64858251   0.52293446]
[ 1.53090411 17.97588093 14.26584774  6.64242918 -5.97716819]

In [111]: #Configuración del gráfico

fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-20,20)
plt.ylim(-20,20)
plt.title("Radar")

plt.hlines(0,-20,20, color="blue")
plt.vlines(0,-20,20, color="blue")

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

#Configuración del color
color = np.where((xnp<= 0) , "red", "blue")
color = np.where((ynp <= 0) , "green", "purple")

plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
plt.plot(xnp, ynp, linestyle="dotted")
plt.plot(xnp, ynp, linestyle="dashdot")
plt.plot(xnp, ynp, linestyle="dashed")
plt.show()

```

También el programa en Python muestra los resultados de las coordenadas de cada objeto, la distancia con respecto al radar, los grados y el objeto. Ver imagen 6.

Imagen 6

Objeto: 1
Distancia en la que se encuentra el objeto: 11
Grados en la que se encuentra el objeto: 8
Coordenadas en el plano cartesiano: [10.892948756157274 , 1.5309041105607197]

Objeto: 2
Distancia en la que se encuentra el objeto: 20
Grados en la que se encuentra el objeto: 64
Coordenadas en el plano cartesiano: [8.76742293578155 , 17.97588092598334]

Objeto: 3
Distancia en la que se encuentra el objeto: 15
Grados en la que se encuentra el objeto: 108
Coordenadas en el plano cartesiano: [-4.63525491562421 , 14.265847744427305]

Objeto: 4
Distancia en la que se encuentra el objeto: 17
Grados en la que se encuentra el objeto: 157
Coordenadas en el plano cartesiano: [-15.648582508691486 , 6.642429184317654]

Objeto: 5
Distancia en la que se encuentra el objeto: 6
Grados en la que se encuentra el objeto: 275
Coordenadas en el plano cartesiano: [0.5229344564859473 , -5.977168188550474]

```
In [112]: #En este segmento de código se despliegan los datos obtenidos desde el puerto serial, objeto, grados, distancia.
#También se imprime las coordenadas cartesianas.
for n in range(0, len(lista), 1):
    print("Objeto: " , objeto[n])
    print("Distancia en la que se encuentra el objeto: " , distancia[n])
    print("Grados en la que se encuentra el objeto: " , grados[n])
    print("Coordenadas en el plano cartesiano: [ " , x[n] , " , " , y[n] , " ]")
    print("\n")
```

Para continuar con el cálculo de las distancias entre los objetos, primero hacer todas las posibles combinaciones que puede tener cada objeto. Para esto creamos algunas listas que ayudaran a la generación de todas las posibilidades. Ver imagen 7

Imagen 7

objetos: [1, 2, 3, 4, 5]
Serie objetos: [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]
combinaciones: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

```
In [113]: #Para calcular la distancia entre dos objetos es importante crear todas las posibles combinaciones entre los objetos.
#Generamos algunas listas para las posibles combinaciones.
objetos=[]
combinaciones=[]
serieObjetos=[]

for n in range(1, len(lista)+1, 1):
    objetos.append(n)

for m in range(1, len(objetos)+1, 1):
    for n in range(1, len(objetos)+1, 1):
        serieObjetos.append(m)

for m in range(1, len(lista)+1, 1):
    for n in range(1, len(lista)+1, 1):
        combinaciones.append(n)

print("objetos: ",objetos)
print("Serie objetos: ", serieObjetos)
print("combinaciones: ",combinaciones)
```

A continuación, llenamos todas las listas () con sus posibilidades. Ver imagen 8.

Imagen 8

```
Combinación objetosXY1 : [[1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]]
Posición objetosXY1 : [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4]
x1: [10.892948756157274, 10.892948756157274, 10.892948756157274, 10.892948756157274, 10.892948756157274, 8.76742293578155, 8.76742293578155, 8.76742293578155, 8.76742293578155, -4.63525491562421, -4.63525491562421, -4.63525491562421, -4.63525491562421, -4.63525491562421, -4.63525491562421, -15.648582508691486, -15.648582508691486, -15.648582508691486, -15.648582508691486, -15.648582508691486, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473, 0.5229344564859473]
y1: [1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 1.5309041105607197, 17.97588092598334, 17.97588092598334, 17.97588092598334, 17.97588092598334, 14.265847744427305, 14.265847744427305, 14.265847744427305, 14.265847744427305, 14.265847744427305, 14.265847744427305, 6.642429184317654, 6.642429184317654, 6.642429184317654, 6.642429184317654, 6.642429184317654, -5.977168188550474, -5.977168188550474, -5.977168188550474, -5.977168188550474, -5.977168188550474]

Combinación objetosXY2 : [[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]]
Posición ObjetosXY2: [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4]
x2: [10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473, 10.892948756157274, 8.76742293578155, -4.63525491562421, -15.648582508691486, 0.5229344564859473]
y2: [1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474, 1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474]
```



```
In [114]: #Para continuar con las posibles combinaciones, Llenamos las listas con todas las posibilidades.
posObjetosXY1=[]
posObjetosXY2=[]
combObjetosXY1=[]
combObjetosXY2=[]
combX1=[]
combX2=[]
combY1=[]
combY2=[]

combDistanciaObjetos=[]

for n in range(0, len(serieObjetos), 1):
    print("Objeto: ",serieObjetos[n],"Posición objeto: ", objeto.index(serieObjetos[n]),"Coordenada x: ",x[objeto.index(serieObjetos[n])])
    posObjetosXY1.append(objeto.index(serieObjetos[n]))
    combX1.append(x[objeto.index(serieObjetos[n])])
    combY1.append(y[objeto.index(serieObjetos[n])])
combObjetosXY1.append(serieObjetos)

print("\n")
for m in range(0, len(combinaciones), 1):
    print("Objeto: ", combinaciones[m],"Posición objeto: ", objeto.index(combinaciones[m]),"Coordenada x: ",x[objeto.index(combinaciones[m])])
    posObjetosXY2.append(objeto.index(combinaciones[m]))
    combX2.append(x[objeto.index(combinaciones[m])])
    combY2.append(y[objeto.index(combinaciones[m])])
combObjetosXY2.append(combinaciones)

print("\n")

print("Combinación objetosXY1 : ",combObjetosXY1)
print("Posición objetosXY1 : ",posObjetosXY1)
print("x1:", combX1)
print("y1:", combY1)
print("\n")

print("Combinación objetosXY2 : ",combObjetosXY2)
print("Posición ObjetosXY2: ",posObjetosXY2)
print("x2:", combX2)
print("y2:", combY2)
```

En el antepenúltimo paso, el programa calcula todas las distancias que puede existir entre los objetos. Ver imagen 9.

Imagen 9

<p>Distancia: 0.0</p> <p>x1: 10.892948756157274 x2: 10.892948756157274</p> <p>y1: 1.5309041105607197 y2: 1.5309041105607197</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p>	<p>Distancia: 16.58177078821413</p> <p>x1: 10.892948756157274 x2: 8.76742293578155</p> <p>y1: 1.5309041105607197 y2: 17.97588092598334</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p> <p>Objeto: 2</p> <p>Posición objeto: 1</p> <p>Coordenada x: 8.76742293578155</p> <p>Coordenada y: 17.97588092598334</p>	<p>Distancia: 20.08242760798821</p> <p>x1: 10.892948756157274 x2: -4.63525491562421</p> <p>y1: 1.5309041105607197 y2: 14.265847744427305</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p> <p>Objeto: 3</p> <p>Posición objeto: 2</p> <p>Coordenada x: -4.63525491562421</p> <p>Coordenada y: 14.265847744427305</p>	<p>Distancia: 27.0292539753244</p> <p>x1: 10.892948756157274 x2: -15.648582508691486</p> <p>y1: 1.5309041105607197 y2: 6.642429184317654</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p> <p>Objeto: 4</p> <p>Posición objeto: 3</p> <p>Coordenada x: -15.648582508691486</p> <p>Coordenada y: 6.642429184317654</p>
<p>Distancia: 12.802669496810145</p> <p>x1: 10.892948756157274 x2: 0.5229344564859473</p> <p>y1: 1.5309041105607197 y2: -5.977168188550474</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p> <p>Objeto: 5</p> <p>Posición objeto: 4</p> <p>Coordenada x: 0.5229344564859473</p> <p>Coordenada y: -5.977168188550474</p>	<p>Distancia: 16.58177078821413</p> <p>x1: 8.76742293578155 x2: 10.892948756157274</p> <p>y1: 17.97588092598334 y2: 1.5309041105607197</p> <p>Objeto: 2</p> <p>Posición objeto: 1</p> <p>Coordenada x: 8.76742293578155</p> <p>Coordenada y: 17.97588092598334</p> <p>Objeto: 1</p> <p>Posición objeto: 0</p> <p>Coordenada x: 10.892948756157274</p> <p>Coordenada y: 1.5309041105607197</p>	<p>Distancia: 0.0</p> <p>x1: 8.76742293578155 x2: 8.76742293578155</p> <p>y1: 17.97588092598334 y2: 17.97588092598334</p> <p>Objeto: 2</p> <p>Posición objeto: 1</p> <p>Coordenada x: 8.76742293578155</p> <p>Coordenada y: 17.97588092598334</p> <p>Objeto: 2</p> <p>Posición objeto: 1</p> <p>Coordenada x: 8.76742293578155</p> <p>Coordenada y: 17.97588092598334</p>	<p>Distancia: 13.906693345177684</p> <p>x1: 8.76742293578155 x2: -4.63525491562421</p> <p>y1: 17.97588092598334 y2: 14.265847744427305</p> <p>Objeto: 2</p> <p>Posición objeto: 1</p> <p>Coordenada x: 8.76742293578155</p> <p>Coordenada y: 17.97588092598334</p> <p>Objeto: 3</p> <p>Posición objeto: 2</p> <p>Coordenada x: -4.63525491562421</p> <p>Coordenada y: 14.265847744427305</p>

<p>Distancia: 26.918180663729892 x1: 8.76742293578155 x2: -15.648582508691486 y1: 17.97588092598334 y2: 6.642429184317654</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>	<p>Distancia: 25.33219596025001 x1: 8.76742293578155 x2: 0.5229344564859473 y1: 17.97588092598334 y2: -5.977168188550474</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p>	<p>Distancia: 20.08242760790821 x1: -4.63525491562421 x2: 10.892948756157274 y1: 14.265847744427305 y2: 1.5309041105607197</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 1 Posición objeto: 0 Coordenada x: 10.892948756157274 Coordenada y: 1.5309041105607197</p>	<p>Distancia: 13.90669345177684 x1: -4.63525491562421 x2: 8.76742293578155 y1: 14.265847744427305 y2: 17.97588092598334</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p>
<p>Distancia: 0.0 x1: -4.63525491562421 x2: -4.63525491562421 y1: 14.265847744427305 y2: 14.265847744427305</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p>	<p>Distancia: 13.394397904155355 x1: -4.63525491562421 x2: -15.648582508691486 y1: 14.265847744427305 y2: 6.642429184317654</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>	<p>Distancia: 20.88986863676606 x1: -4.63525491562421 x2: 0.5229344564859473 y1: 14.265847744427305 y2: -5.977168188550474</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p>	<p>Distancia: 27.0292539753244 x1: -15.648582508691486 x2: 10.892948756157274 y1: 6.642429184317654 y2: 1.5309041105607197</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p> <p>Objeto: 1 Posición objeto: 0 Coordenada x: 10.892948756157274 Coordenada y: 1.5309041105607197</p>
<p>Distancia: 26.918180663729892 x1: -15.648582508691486 x2: 8.76742293578155 y1: 6.642429184317654 y2: 17.97588092598334</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p>	<p>Distancia: 13.394397904155355 x1: -15.648582508691486 x2: -4.63525491562421 y1: 6.642429184317654 y2: 14.265847744427305</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p>	<p>Distancia: 0.0 x1: -15.648582508691486 x2: -15.648582508691486 y1: 6.642429184317654 y2: 6.642429184317654</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>	<p>Distancia: 20.512732602174726 x1: -15.648582508691486 x2: 0.5229344564859473 y1: 6.642429184317654 y2: -5.977168188550474</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p>
<p>Distancia: 12.802669496018145 x1: 0.5229344564859473 x2: 10.892948756157274 y1: -5.977168188550474 y2: 1.5309041105607197</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p> <p>Objeto: 1 Posición objeto: 0 Coordenada x: 10.892948756157274 Coordenada y: 1.5309041105607197</p>	<p>Distancia: 25.33219596025001 x1: 0.5229344564859473 x2: 8.76742293578155 y1: -5.977168188550474 y2: 17.97588092598334</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p>	<p>Distancia: 20.88986863676606 x1: 0.5229344564859473 x2: -4.63525491562421 y1: -5.977168188550474 y2: 14.265847744427305</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p>	<p>Distancia: 20.512732602174726 x1: 0.5229344564859473 x2: -15.648582508691486 y1: -5.977168188550474 y2: 6.642429184317654</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>

```

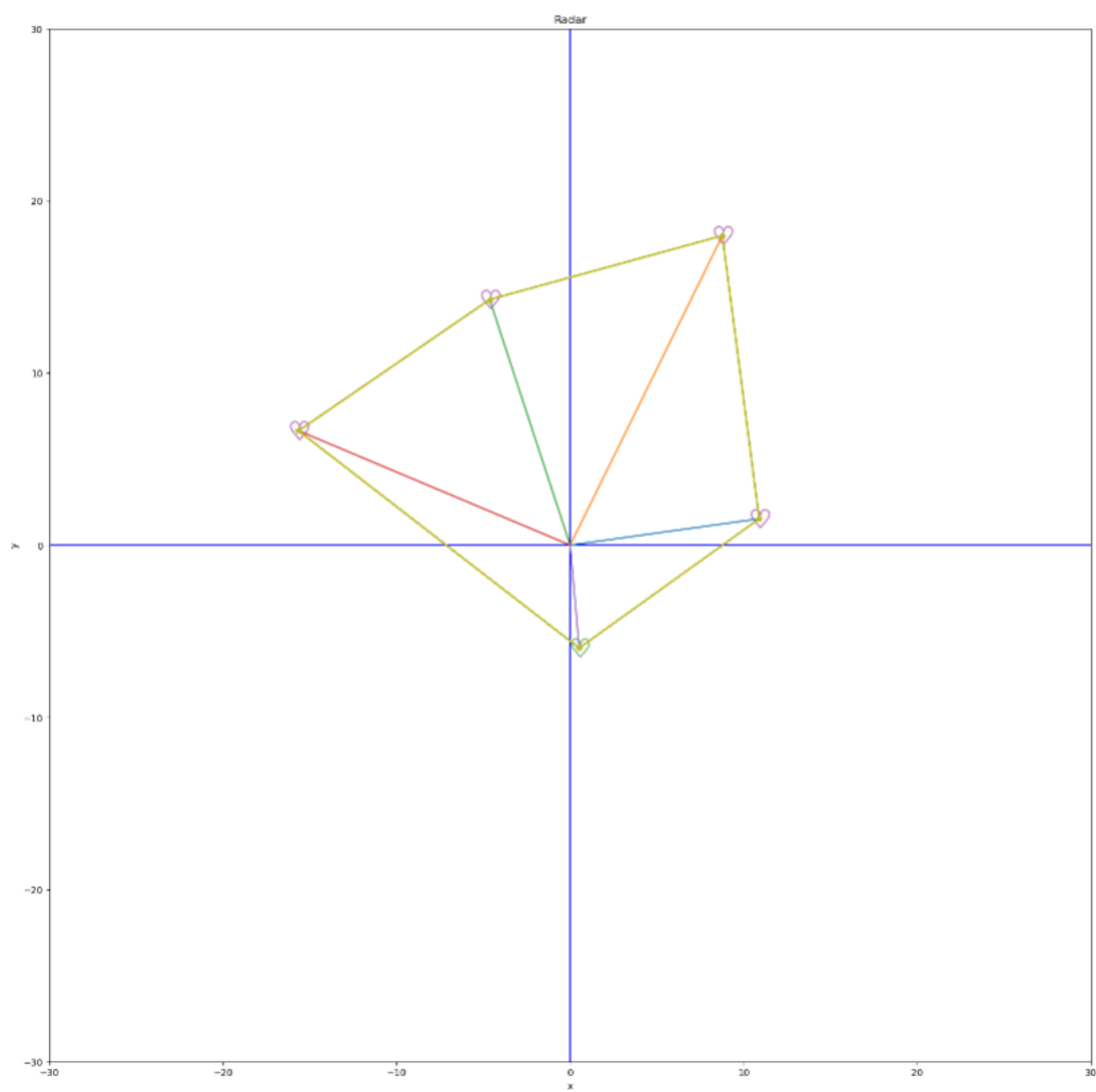
In [116]: #En este segmento de código se calcula la distancia de todas las posibles combinaciones entre Los objetos y
#se despliegan los resultados.
print(objetos)
print(x)
print(y)
print("\n")
print("\n")
for m in range(0, len(combinaciones), 1):
    print("\n")
    print("\n")
    combDistanciaObjetos.append(float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)))
    print("Distancia: ", float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)) )
    print("x1: ",combX1[m], " x2:", combX2[m], )
    print("y1: ",combY1[m], " y2:",combY2[m])
    print("\n")
    print("Objeto: ",serieObjetos[m])
    print("Posición objeto: ", objeto.index(serieObjetos[m]))
    print("Coordenada x: ",x[objeto.index(serieObjetos[m])])
    print("Coordenada y: ",y[objeto.index(serieObjetos[m])])
    print("\n")
    print("Objeto: ", combinaciones[m])
    print("Posición objeto: ", objeto.index(combinaciones[m]))
    print("Coordenada x: ",x[objeto.index(combinaciones[m])])
    print("Coordenada y:",y[objeto.index(combinaciones[m])])

print(combDistanciaObjetos)
print("\n")

```

Por último, volvemos a graficar los objetos en el plano cartesiano. Ver imagen 10.

Imagen 10



In [117]: *#Volvemos a graficar.*

```
fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-30,30)
plt.ylim(-30,30)
plt.title("Radar")

plt.hlines(0,-30,30, color="blue")
plt.vlines(0,-30,30, color="blue")

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

#Configuración del color
color = np.where((xnp<= 0) , "red", "blue")
color = np.where((ynp <= 0) , "green", "purple")

plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
ax.plot(combX1,comby1, marker='*')
ax.plot(combX2,comby2, marker='*')

ax.plot(combX1,comby1, marker='*')
ax.plot(combX2,comby2, marker='*')
plt.show()
```

Código

```
##Autor Plata Luna Iveth Vanessa

##Importamos la librería para el uso del puerto serial y time par el manejo del tiempo.

import serial

import time

##El programa se conecta al puerto serial COM6 a 9600 baudios dando tiempos de demora para
recibir datos.

serialArduino= serial.Serial("COM6", 9600 )

time.sleep(1)

#La variable cad almacenará de manera temporal los datos obtenidos del puerto serial.

cad = ""

#Definimos una lista para almacenar los datos del puerto serial. La variable i es de control.

lista=[]

i = 0

#Segmento de código que obtiene datos desde el puerto serial y lo transforma a ASCII. El bucle
termina hasta que se obtene una 'f'

while not(cad[0:1] == 'f') :

    cad = serialArduino.readline().decode('ascii')

    print(cad)

    if not(cad[0:1] == 'f'):

        lista.append(cad)

        i=i+1

print(cad)

print(lista)

print(i)

#La definición de estas listas es para almacenar los datos obtenidos del puerto serial,
previamente se decodifican.
```

Elaboró: Plata Luna Iveth Vanessa

```

objeto=[]
distancia=[]
grados=[]

#Definimos listas que servirán a decodificar la información del puerto serial.
index0 = []
indexo= []
indexD = []
indexd = []
indexG = []
indexg = []
numDig0 = []
numDigD = []
numDigG = []
objetoS=[]
distanciaS=[]
gradosS=[]

#Segmento de código que decodifica las cadenas de caracter obtenidas desde el puerto serial.
#Se utiliza funciones para la manipulación de cadenas, como buscar un caracter y extraer
subcadenas.
#Los resultados obtenidos se almacenan en listas.
for n in range(0, len(lista), 1):
    print(lista[n])
    cadena=lista[n]

    index0.append(cadena.find('R'))
    indexo.append(cadena.find('r'))
    indexD.append(cadena.find('D'))
    indexd.append(cadena.find('d'))
    indexG.append(cadena.find('G'))
    indexg.append(cadena.find('g'))

    numDig0.append (indexo[n] - index0[n] - 1)
    numDigD.append (indexd[n] - indexD[n] - 1)

```

```

numDigG.append (indexg[n] - indexG[n] - 1)

objetoS.append(cadena[(index0[n]+1):(index0[n]+1+numDig0[n]]))
distanciaS.append(cadena[(indexD[n]+1):(indexD[n]+1+numDigD[n]]))
gradosS.append(cadena[(indexG[n]+1):(indexG[n]+1+numDigG[n]]))

print("índice de caracteres")
print(index0)
print(indexo)
print(indexD)
print(indexd)
print(indexG)
print(indexg)
print("Número de dígitos")
print(numDig0)
print(numDigD)
print(numDigG)
print("Resultados")
print(objetoS)
print(distanciaS)
print(gradosS)

#En este segmento de código se convierten las coordenadas polares a coordenadas cartesianas.
#Para esta función utilizamos la libreria math.

import math
objeto = []
distancia = []
grados = []

for n in range(0, len(lista), 1):

    objeto.append (int (objetoS[n]))
    distancia.append (int (distanciaS[n]))
    grados.append(int (gradosS[n]))

```

```

x=[]
y=[]

for n in range(0, len(lista), 1):
    x.append(distancia[n] * math.cos(math.radians(grados[n])));
    y.append(distancia[n] * math.sin(math.radians(grados[n])));

print(objeto)
print(distancia)
print(grados)
print(x)
print(y)

#Para graficar utilizamos las librerias mumpy y matplotlib.
import numpy as np
import matplotlib.pyplot as plt

#Es importante convertir la lista a un tipo arreglo para poder graficar.
xnp = np.array(x)
ynp = np.array(y)
print(xnp)
print(ynp)

#Configuración del gráfico

fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-20,20)
plt.ylim(-20,20)
plt.title("Radar")

plt.hlines(0,-20,20, color="blue")
plt.vlines(0,-20,20, color="blue")

```



```

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]], [0,y[n]])

#Configuración del color
color = np.where((xnp<= 0) , "red", "blue")
color = np.where((ynp <= 0) , "green", "purple")

plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
plt.plot(xnp, ynp, linestyle="dotted")
plt.plot(xnp, ynp, linestyle="dashdot")
plt.plot(xnp, ynp, linestyle="dashed")
plt.show()

#En este segmento de código se despliegan los datos obtenidos desde el puerto serial, objeto,
grados, distancia.

#También se imprime las coordenadas cartesianas.
for n in range(0, len(lista), 1):
    print("Objeto: " , objeto[n])
    print("Distancia en la que se encuentra el objeto: " , distancia[n])
    print("Grados en la que se encuentra el objeto: " , grados[n])
    print("Coordenadas en el plano cartesiano: [ " , x[n] , " , " , y[n] , " ]")
    print("\n")

#Para calcular la distancia entre dos objetos es importante crear todas las posibles
combinaciones entre los objetos.

#Generamos algunas listas para las posibles combinaciones.
objetos=[]
combinaciones=[]
serieObjetos=[]

for n in range(1, len(lista)+1, 1):
    objetos.append(n)

```

```

for m in range(1, len(objetos)+1, 1):
    for n in range(1, len(objetos)+1, 1):
        serieObjetos.append(m)

for m in range(1, len(lista)+1, 1):
    for n in range(1, len(lista)+1, 1):
        combinaciones.append(n)

print("objetos: ",objetos)
print("Serie objetos: ", serieObjetos)
print("combinaciones: ",combinaciones)

#Para continuar con las posibles combinaciones, llenamos las listas con todas las
posibilidades.

posObjetosXY1=[]
posObjetosXY2=[]
combObjetosXY1=[]
combObjetosXY2=[]

combX1=[]
combX2=[]
combY1=[]
combY2=[]

combDistanciaObjetos=[]

for n in range(0, len(serieObjetos), 1):
    print("Objeto: ",serieObjetos[n],"Posición objeto: ",
objeto.index(serieObjetos[n]),"Coordenada x: ",x[objeto.index(serieObjetos[n])],"Coordenada
y: ",y[objeto.index(serieObjetos[n])])
    posObjetosXY1.append(objeto.index(serieObjetos[n]))
    combX1.append(x[objeto.index(serieObjetos[n])])
    combY1.append(y[objeto.index(serieObjetos[n])])
    combObjetosXY1.append(serieObjetos)

```

```

print("\n")
for m in range(0, len(combinaciones), 1):
    print("Objeto: ", combinaciones[m], "Posición objeto: ",
    objeto.index(combinaciones[m]), "Coordenada x: ", x[objeto.index(combinaciones[m])], "Coordenada
    y:", y[objeto.index(combinaciones[m])])

    posObjetosXY2.append(objeto.index(combinaciones[m]))

    combX2.append(x[objeto.index(combinaciones[m])])
    combY2.append(y[objeto.index(combinaciones[m])])
combObjetosXY2.append(combinaciones)

print("\n")

print("Combinación objetosXY1 : ", combObjetosXY1)
print("Posición objetosXY1 : ", posObjetosXY1)
print("x1:", combX1)
print("y1:", combY1)
print("\n")

print("Combinación objetosXY2 : ", combObjetosXY2)
print("Posición ObjetosXY2: ", posObjetosXY2)
print("x2:", combX2)
print("y2:", combY2)

#En este segmento de código se calcula la distancia de todas las posibles combinaciones entre
los objetos y
#se despliegan los resultados.
print(objetos)
print(x)
print(y)
print("\n")
print("\n")

for m in range(0, len(combinaciones), 1):
    print("\n")
    print("\n")
    combDistanciaObjetos.append(float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-
    combY2[m])**2)))
    print("Distancia: ", float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)) )

```

```

print("x1: ",combX1[m]," x2:", combX2[m], )
print("y1: ",combY1[m]," y2:",combY2[m])
print("\n")
print("Objeto: ",serieObjetos[m])
print("Posición objeto: ", objeto.index(serieObjetos[m]))
print("Coordenada x: ",x[objeto.index(serieObjetos[m])])
print("Coordenada y: ",y[objeto.index(serieObjetos[m])])
print("\n")
print("Objeto: ", combinaciones[m])
print("Posición objeto: ", objeto.index(combinaciones[m]))
print("Coordenada x: ",x[objeto.index(combinaciones[m])])
print("Coordenada y:",y[objeto.index(combinaciones[m])])

print(combDistanciaObjetos)
print("\n")

#Volvemos a graficar.

fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-30,30)
plt.ylim(-30,30)
plt.title("Radar")

plt.hlines(0,-30,30, color="blue")
plt.vlines(0,-30,30, color="blue")

for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

```

```

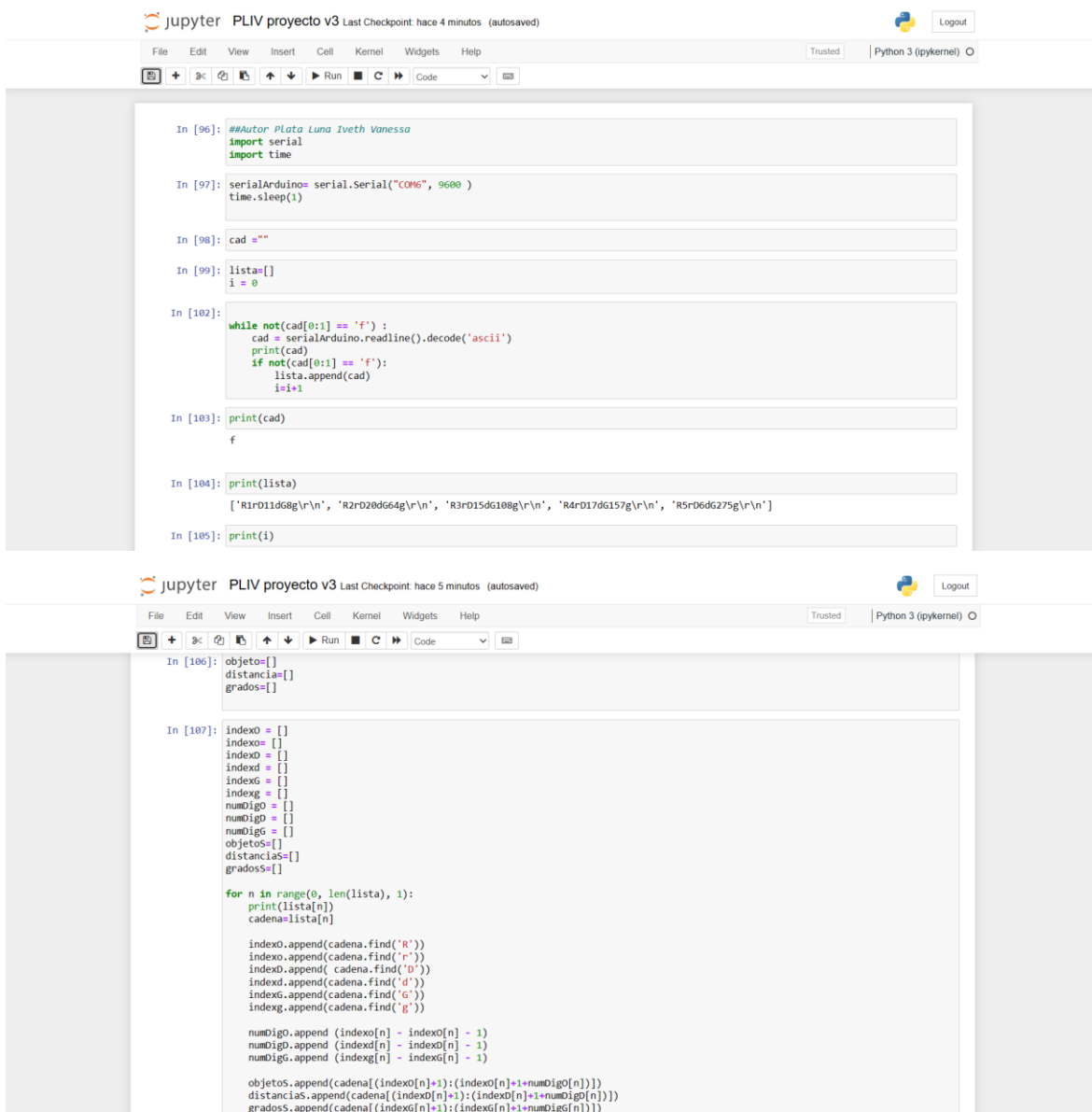
#Configuración del color
color = np.where((xnp<= 0) , "red", "blue")
color = np.where((ynp <= 0) , "green", "purple")

plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuit$', alpha=0.4 )
ax.plot(combX1,comby1, marker='*')
ax.plot(combX2,comby2, marker='*')

ax.plot(combX1,comby1, marker='*')
ax.plot(combX2,comby2, marker='*')
plt.show()

```

Anexo



The image displays two screenshots of a Jupyter Notebook interface. The top screenshot shows the first five code cells (In [96] to In [105]). The bottom screenshot shows the next two code cells (In [106] and In [107]). The notebook is titled 'PLIV proyecto v3' and shows a 'Last Checkpoint' of 4 minutes. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code is written in Python 3 (ipykernel).

```
In [96]: ##Autor: Plata Luna Iveth Vanessa
import serial
import time

In [97]: serialArduino= serial.Serial("COM6", 9600 )
time.sleep(1)

In [98]: cad =""

In [99]: lista=[]
i = 0

In [102]: while not(cad[0:1] == 'f') :
cad = serialArduino.readline().decode('ascii')
print(cad)
if not(cad[0:1] == 'f'):
lista.append(cad)
i=i+1

In [103]: print(cad)
f

In [104]: print(lista)
['R1rD11dG8g\r\n', 'R2rD20dG64g\r\n', 'R3rD15dG108g\r\n', 'R4rD17dG157g\r\n', 'R5rD6dG275g\r\n']

In [105]: print(i)
```

```
In [106]: objeto=[]
distancia=[]
grados=[]

In [107]: index0 = []
indexo = []
indexD = []
indexd = []
indexG = []
indexg = []
numDigo = []
numDigD = []
numDigG = []
objetos=[]
distancias=[]
grados=[]

for n in range(0, len(lista), 1):
print(lista[n])
cadena=lista[n]

index0.append(cadena.find('R'))
indexo.append(cadena.find('r'))
indexD.append(cadena.find('D'))
indexd.append(cadena.find('d'))
indexG.append(cadena.find('G'))
indexg.append(cadena.find('g'))

numDigo.append (index0[n] - indexo[n] - 1)
numDigD.append (indexD[n] - indexo[n] - 1)
numDigG.append (indexG[n] - indexg[n] - 1)

objetos.append(cadena[(index0[n]+1):(index0[n]+1+numDigo[n])])
distancias.append(cadena[(indexD[n]+1):(indexD[n]+1+numDigD[n])])
grados.append(cadena[(indexG[n]+1):(indexG[n]+1+numDigG[n])])
```

Python 3 (pykernel)

Logout

FileEditViewInsertCellKernelWidgetsHelp

RunCode

move selected cells up

```
print("Indice de caracteres")
print(index0)
print(index1)
print(index2)
print(index3)
print(index4)
print(index5)
print(index6)
print(index7)
print("Número de dígitos")
print(numDig0)
print(numDig1)
print(numDig2)
print("Resultados")
print(objetos)
print(distancias)
print(grados5)

R1rD11dG8g

R2rD20dG64g

R3rD15dG100g

R4rD17dG157g

R5rD6dG275g

índice de caracteres
[0, 0, 0, 0, 0]
[2, 2, 2, 2, 2]
[3, 3, 3, 3, 3]
[6, 6, 6, 6, 6]
[7, 7, 7, 7, 6]
[9, 10, 11, 11, 10]
Número de dígitos
[1, 1, 1, 1, 1]
[2, 2, 2, 2, 1]
[1, 2, 3, 3, 3]
```

jupyter PLIV proyecto v3 Last Checkpoint: hace 7 minutos (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```

In [110]: xnp = np.array(x)
          ynp = np.array(y)
          print(xnp)
          print(ynp)

[ 10.89294876  8.76742294 -4.63525492 -15.64858251  0.52293446]
[ 1.53090411 17.97588093 14.26584774  6.64242918 -5.97716819]

In [111]: #configuración del gráfico

fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-20,20)
plt.ylim(-20,20)
plt.title("Radar")

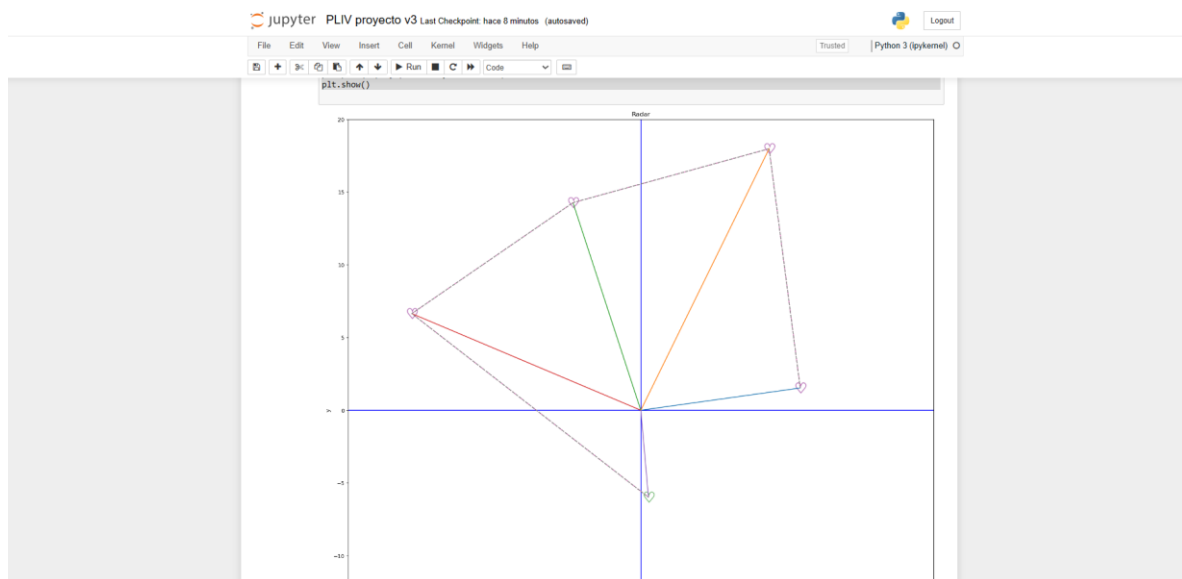
plt.hlines(0,-20,20, color="blue")
plt.vlines(0,-20,20, color="blue")

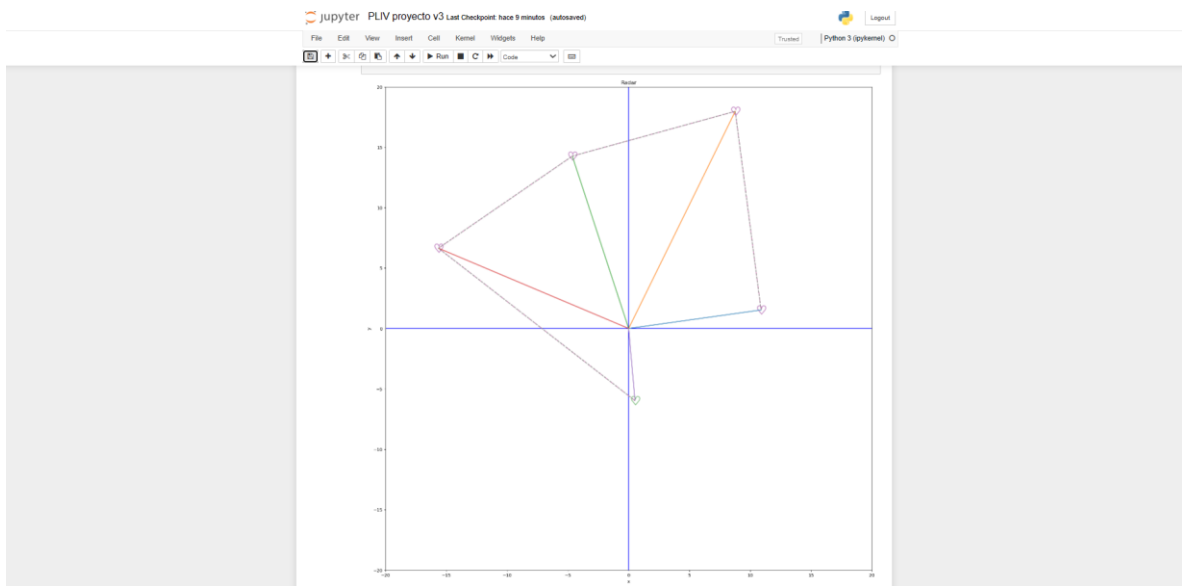
for n in range(0, len(lista), 1):
    plt.plot([0,x[n]],[0,y[n]])

#configuración del color
color = np.where((xnp<= 0) , "red", "blue")
color = np.where((ynp <= 0) , "green", "purple")

plt.scatter(xnp, ynp, c=color, label=color, s=500, marker=r'$\heartsuits$', alpha=0.4 )
plt.plot(xnp, ynp, linestyle="dotted")
plt.plot(xnp, ynp, linestyle="dashdot")
plt.plot(xnp, ynp, linestyle="dashed")

```





Jupyter PLIV proyecto v3 Last Checkpoint: hace 9 minutos (autosaved)

```
In [112]: for n in range(0, len(lista), 1):
           print("Objeto: ", objeto[n])
           print("Distancia en la que se encuentra el objeto: ", distancia[n])
           print("Grados en la que se encuentra el objeto: ", grados[n])
           print("Coordenadas en el plano cartesiano: [ " , x[n] , " , " , y[n] , " ]")
           print("\n")
```

Objeto: 1
Distancia en la que se encuentra el objeto: 11
Grados en la que se encuentra el objeto: 8
Coordenadas en el plano cartesiano: [10.892948756157274 , 1.5309041105607197]

Objeto: 2
Distancia en la que se encuentra el objeto: 20
Grados en la que se encuentra el objeto: 64
Coordenadas en el plano cartesiano: [8.76742293578155 , 17.97588092598334]

Objeto: 3
Distancia en la que se encuentra el objeto: 15
Grados en la que se encuentra el objeto: 108
Coordenadas en el plano cartesiano: [-4.63525491562421 , 14.265847744427305]

Objeto: 4
Distancia en la que se encuentra el objeto: 17
Grados en la que se encuentra el objeto: 157
Coordenadas en el plano cartesiano: [-15.648582508691486 , 6.642429184317654]

Objeto: 5
Distancia en la que se encuentra el objeto: 6
Grados en la que se encuentra el objeto: 275
Coordenadas en el plano cartesiano: [0.5229344564859473 , -5.977168188558474]

```
jupyter PLIV proyecto v3 Last Checkpoint: hace 10 minutos (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel) Logout

In [113]: objetos=[]
combinaciones=[]
serieObjetos=[]
#comb1=[]
#comb2=[]

for n in range(1, len(lista)+1, 1):
    objetos.append(n)

for m in range(1, len(objetos)+1, 1):
    for n in range(1, len(objetos)+1, 1):
        serieObjetos.append(m)

for m in range(1, len(lista)+1, 1):
    for n in range(1, len(lista)+1, 1):
        combinaciones.append(n)

print("objetos: ",objetos)
print("Serie objetos: ", serieObjetos)
print("combinaciones: ",combinaciones)

objetos: [1, 2, 3, 4, 5]
Serie objetos: [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]
combinaciones: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

In [114]: posObjetosXY1=[]
posObjetosXY2=[]
combObjetosXY1=[]
combObjetosXY2=[]
combX1=[]
combX2=[]
combY1=[]
combY2=[]

combDistanciaObjetos=[]
```

```
jupyter PLIV proyecto v3 Last Checkpoint: hace 10 minutos (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel) Logout

combDistanciaObjetos=[]

for n in range(0, len(serieObjetos), 1):
    print("Objeto: ", serieObjetos[n], "Posición objeto: ", objeto.index(serieObjetos[n]), "Coordenada x: ",x[objeto.index(serieObjetos[n])])
    posObjetosXY1.append(objeto.index(serieObjetos[n]))
    combX1.append(x[objeto.index(serieObjetos[n])])
    combY1.append(y[objeto.index(serieObjetos[n])])
    combObjetosXY1.append(serieObjetos[n])

print("\n")
for m in range(0, len(combinaciones), 1):
    print("Objeto: ", combinaciones[m], "Posición objeto: ", objeto.index(combinaciones[m]), "Coordenada x: ",x[objeto.index(combinaciones[m])])
    posObjetosXY2.append(objeto.index(combinaciones[m]))
    combX2.append(x[objeto.index(combinaciones[m])])
    combY2.append(y[objeto.index(combinaciones[m])])
    combObjetosXY2.append(combinaciones[m])

print("\n")
print("Combinación objetosXY1 : ",combObjetosXY1)
print("Posición objetosXY1 : ",posObjetosXY1)
print("x1:", combX1)
print("y1:", combY1)
print("\n")

print("Combinación objetosXY2 : ",combObjetosXY2)
print("Posición objetosXY2 : ",posObjetosXY2)
print("x2:", combX2)
print("y2:", combY2)
```

[illegible][illegible]

```
jupyter PLIV proyecto v3 Last Checkpoint: hace unos segundos (autosaved)
File Edit View Insert Cell Kernel Widgets Help
+ + + + + Run Code
In [116]: print(objetos)
print(x)
print(y)
print("\n")
print("\n")
for m in range(0, len(combinaciones), 1):
    print("\n")
    combDistanciaObjetos.append(float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)))
    print("Distancia: ", float(math.sqrt((combX1[m]-combX2[m])**2+(combY1[m]-combY2[m])**2)))
    print("x1: ", combX1[m], " x2: ", combX2[m], )
    print("y1: ", combY1[m], " y2: ", combY2[m])
    print("\n")
    print("Objeto: ", serieObjetos[m])
    print("Posición objeto: ", objeto.index(serieObjetos[m]))
    print("Coordenada x: ", x[objeto.index(serieObjetos[m])])
    print("Coordenada y: ", y[objeto.index(serieObjetos[m])])
    print("\n")
    print("Objeto: ", combinaciones[m])
    print("Posición objeto: ", objeto.index(combinaciones[m]))
    print("Coordenada x: ", x[objeto.index(combinaciones[m])])
    print("Coordenada y: ", y[objeto.index(combinaciones[m])])

print(combDistanciaObjetos)
print("\n")
```

```
jupyter PLIV proyecto v3 Last Checkpoint: hace un minuto (autosaved)
File Edit View Insert Cell Kernel Widgets Help
+ + + + + Run Code
print("Objeto: ", combinaciones[m])
print("Posición objeto: ", objeto.index(combinaciones[m]))
print("Coordenada x: ", x[objeto.index(combinaciones[m])])
print("Coordenada y: ", y[objeto.index(combinaciones[m])])

print(combDistanciaObjetos)
print("\n")

[1, 2, 3, 4, 5]
[10.892948756157274, 8.76742293578155, -4.63525491563421, -15.648582508691486, 0.5229344564859473]
[1.5309041105607197, 17.97588092598334, 14.265847744427305, 6.642429184317654, -5.977168188550474]

Distancia: 0.0
x1: 10.892948756157274 x2: 10.892948756157274
y1: 1.5309041105607197 y2: 1.5309041105607197

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

In [117]: #Configuración del gráfico
fig, ax = plt.subplots(figsize = (20,20))

plt.xlabel("x")
plt.ylabel("y")
plt.xlim(-30,30)
plt.ylim(-30,30)
plt.title("Radar")
```

```
Distancia: 0.0
x1: 10.892948756157274 x2: 10.892948756157274
y1: 1.5309041105607197 y2: 1.5309041105607197

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Distancia: 16.58177078821413
x1: 10.892948756157274 x2: 8.76742293578155
y1: 1.5309041105607197 y2: 17.97588092598334

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334
```

```
Distancia: 20.08242760798821
x1: 10.892948756157274 x2: -4.63525491562421
-y1: -1.5309041105607197 -y2: -14.265847744427305
> unscroll output; double click to hide

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305
```

```
Distancia: 27.0292539753244
x1: 10.892948756157274 x2: -15.648582508691486
y1: 1.5309041105607197 y2: 6.642429184317654
```

```
Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197
```

```
Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654
```

```
Distancia: 12.802669496010145
x1: 10.892948756157274 x2: 0.5229344564859473
y1: 1.5309041105607197 y2: -5.977168188550474
```

```
Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197
```

```
Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
```

```
Distancia: 16.58177078821413
x1: 8.76742293578155 x2: 10.892948756157274
y1: 17.97588092598334 y2: 1.5309041105607197
```

```
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334
```

```
Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197
```

```
Distancia: 0.0
x1: 8.76742293578155 x2: 8.76742293578155
y1: 17.97588092598334 y2: 17.97588092598334
```

```
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334
```

```
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334
```

```
Distancia: 13.906693345177684
x1: 8.76742293578155 x2: -4.63525491562421
y1: 17.97588092598334 y2: 14.265847744427305
```

```
Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334
```

```
Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305
```

<p>Distancia: 26.918180663729892 x1: 8.76742293578155 x2: -15.648582508691486 y1: 17.97588092598334 y2: 6.642429184317654</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>	
<p>Distancia: 25.33219596025001 x1: 8.76742293578155 x2: 0.5229344564859473 y1: 17.97588092598334 y2: -5.977168188550474</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p> <p>Objeto: 5 Posición objeto: 4 Coordenada x: 0.5229344564859473 Coordenada y: -5.977168188550474</p>	
<p>Distancia: 20.00242760798821 x1: -4.63525491562421 x2: 10.892948756157274 y1: 14.265847744427305 y2: 1.5309041105607197</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 1 Posición objeto: 0 Coordenada x: 10.892948756157274 Coordenada y: 1.5309041105607197</p>	
<p>Distancia: 13.906693345177684 x1: -4.63525491562421 x2: 8.76742293578155 y1: 14.265847744427305 y2: 17.97588092598334</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 2 Posición objeto: 1 Coordenada x: 8.76742293578155 Coordenada y: 17.97588092598334</p>	
<p>Distancia: 0.0 x1: -4.63525491562421 x2: -4.63525491562421 y1: 14.265847744427305 y2: 14.265847744427305</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p>	
<p>Distancia: 13.394397904155355 x1: -4.63525491562421 x2: -15.648582508691486 y1: 14.265847744427305 y2: 6.642429184317654</p> <p>Objeto: 3 Posición objeto: 2 Coordenada x: -4.63525491562421 Coordenada y: 14.265847744427305</p> <p>Objeto: 4 Posición objeto: 3 Coordenada x: -15.648582508691486 Coordenada y: 6.642429184317654</p>	

Distancia: 20.8896863676606
x1: -4.63525491562421 x2: 0.5229344564859473
y1: 14.265847744427305 y2: -5.977168188550474

Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Distancia: 27.0292539753244
x1: -15.648582508691486 x2: 10.892948756157274
y1: 6.642429184317654 y2: 1.5309041105607197

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Distancia: 26.918180663729892
x1: -15.648582508691486 x2: 8.76742293578155
y1: 6.642429184317654 y2: 17.97588092598334

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334

Distancia: 13.394397904155355
x1: -15.648582508691486 x2: -4.63525491562421
y1: 6.642429184317654 y2: 14.265847744427305

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305

Distancia: 0.0
x1: -15.648582508691486 x2: -15.648582508691486
y1: 6.642429184317654 y2: 6.642429184317654

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Distancia: 20.512732602174726
x1: -15.648582508691486 x2: 0.5229344564859473
y1: 6.642429184317654 y2: -5.977168188550474

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Distancia: 12.802669496010145
x1: 0.5229344564859473 x2: 10.892948756157274
y1: -5.977168188550474 y2: 1.5309041105607197

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Objeto: 1
Posición objeto: 0
Coordenada x: 10.892948756157274
Coordenada y: 1.5309041105607197

Distancia: 25.33219596025001
x1: 0.5229344564859473 x2: 8.76742293578155
y1: -5.977168188550474 y2: 17.97588092598334

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Objeto: 2
Posición objeto: 1
Coordenada x: 8.76742293578155
Coordenada y: 17.97588092598334

Distancia: 20.88986863676606
x1: 0.5229344564859473 x2: -4.63525491562421
y1: -5.977168188550474 y2: 14.265847744427305

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Objeto: 3
Posición objeto: 2
Coordenada x: -4.63525491562421
Coordenada y: 14.265847744427305

Distancia: 20.512732602174726
x1: 0.5229344564859473 x2: -15.648582508691486
y1: -5.977168188550474 y2: 6.642429184317654

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Objeto: 4
Posición objeto: 3
Coordenada x: -15.648582508691486
Coordenada y: 6.642429184317654

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474

Objeto: 5
Posición objeto: 4
Coordenada x: 0.5229344564859473
Coordenada y: -5.977168188550474
[0.0, 16.58177078821413, 20.08242760798821, 27.0292539753244, 12.802669496010145, 16.58177078821413, 0.0, 13.906693345177684, 26.918180663729892, 25.33219596025001, 20.08242760798821, 13.906693345177684, 0.0, 13.94397904155355, 20.88986863676606, 27.0292539753244, 26.918180663729892, 13.94397904155355, 0.0, 20.512732602174726, 12.802669496010145, 25.33219596025001, 20.88986863676606, 20.512732602174726, 0.0, 0.0, 16.58177078821413, 20.08242760798821, 27.0292539753244, 12.802669496010145, 16.58177078821413, 0.0, 13.906693345177684, 26.918180663729892, 25.33219596025001, 20.08242760798821, 13.906693345177684, 0.0, 13.94397904155355, 20.88986863676606, 27.0292539753244, 26.918180663729892, 13.94397904155355, 0.0, 20.512732602174726, 12.802669496010145, 25.33219596025001, 20.88986863676606, 20.512732602174726, 0.0]

