Gợi ý lập trình Robot tự động VGU ROBOCON 2019

Trương Minh Hiếu
Nguyễn Hồng Lam Giang
28th April 2019

OUTLINE

Lời giới thiệu

Xin chào các bạn, đây là tài liệu của BTC VGU Robocon 2019 nhằm gợi ý các bạn chế tạo con Robot tự động. Bản Source Code kèm theo sẽ đưa ra một khung sườn đầy đủ từ việc lấy data từ Server, xử lý thuật toán đến việc gửi truyền nhận tín hiệu cho Robot. Tuy nhiên, để tránh trùng lặp với những tài liệu đã gửi trước đó, thì tài liệu này sẽ tập trung giải quyết hai vấn đề:

- Kết nối và điều khiển Robot bằng Laptop
- Gợi ý thuật toán điều khiển cơ bản

Bản gợi ý này tuy không phải là một bản đã được tối ưu hóa về mọi mặt, phần điều khiển Robot cũng rất là đơn giản, tuy nhiên nó đủ để đưa ra cho các bạn một khung sườn khá là trực quan để cho các bạn có thể tùy thích phát triển lên và tối ưu hóa mọi thứ theo ý thích của mình.

Mục tiêu của tài liệu này chỉ dừng ở mức chế tạo ra một con robot tự động đi tới một object có sẵn trên sân, quay đầu và mang object đó trở về khu vực storage zone.

Các bạn lưu ý rằng để hiểu tài liệu dễ dàng hơn thì mình coi thêm file source code đính kèm, bao gồm **dataProcessing.py** - việc xử lý data, **AutobotSample.py** - nhận data từ server và chạy Robot.

Ah, còn một lưu ý nhỏ nữa, bởi vì cuộc thi của mình cũng có những bạn cấp 3 tham gia, cũng chưa có kinh nghiệm nhiều. Hoặc cũng một phần là thời gian cuộc thi hơi gấp rút, nên chưa kịp hoàn thiện robot tự động. Các bạn hoàn toàn có thể chỉ cần chế tạo một con điều khiển bằng tay và đem đi thi luôn nhé. Hay cho nó chạy độc lập không cần dữ liệu từ server BTC cũng được.

Chúc bạn một cuộc thi thật trọn vẹn và tràn đầy niềm vui,

BTC VGU ROBOCON 2019: Battle Mining Bots

I. Cấu tạo và chức năng Robot tự động:

a) Các bộ phận chính

- Board Arduino Nano
- Module Bluetooth HC-05 + Mạch chuyển mức điện áp giao tiếp 5v - 3.3v
- Mạch điều khiển động cơ DC L298

b) Chức năng

Vì mục đích tài liệu này chỉ là gợi ý, nên tụi mình sẽ không can thiệp quá sâu vào phần lập trình Robot.

Con Robot tự động trong tài liệu này chỉ thực hiện được bốn chức năng đơn giản:

- Đi thẳng
- Quẹo phải
- Dừng
- Thay đổi tốc độ bằng biến trở

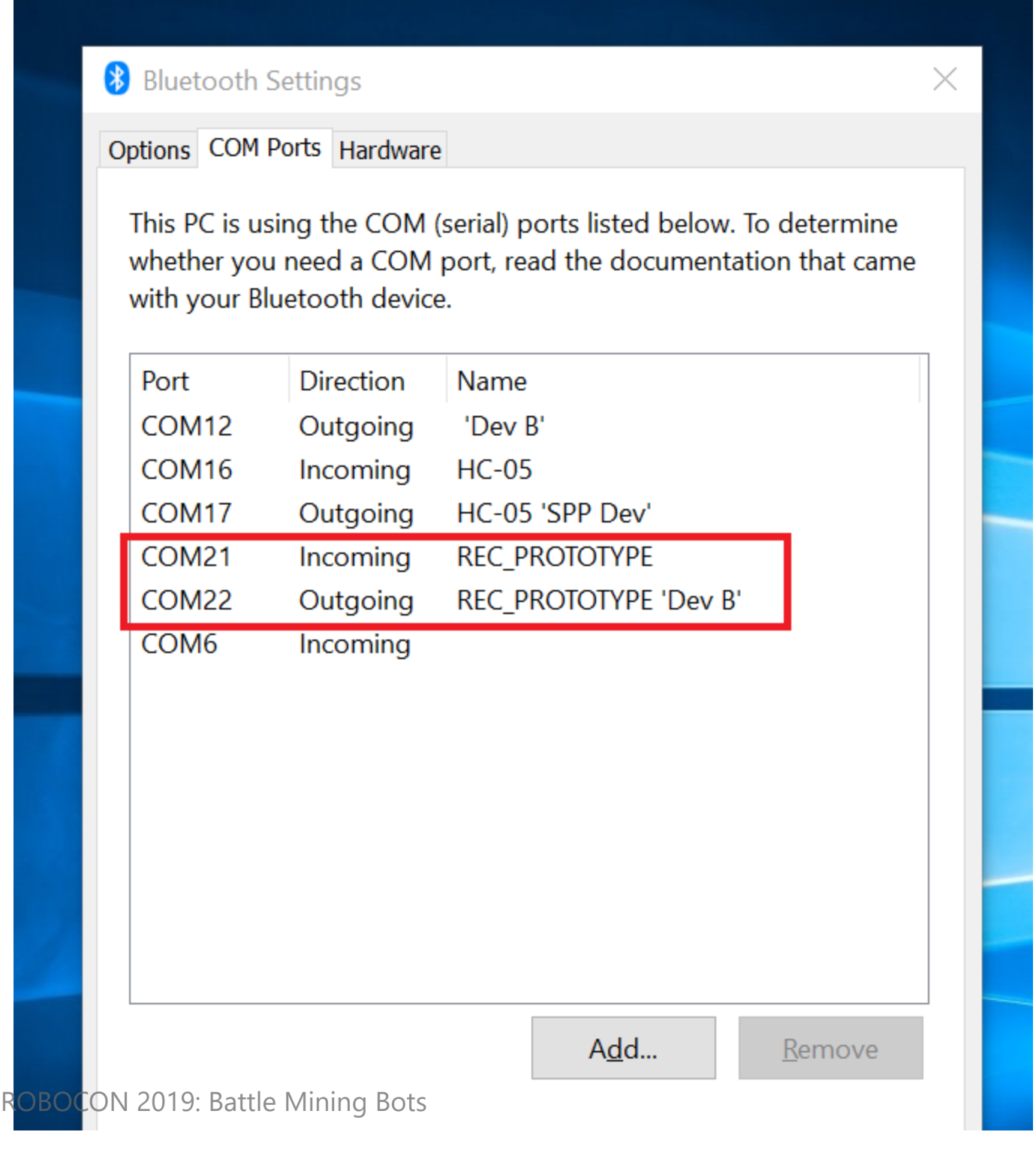
II. Các hàm cần chuẩn bị

1 Kết nối với Robot thông qua thư viện pySerial

Để kết nối từ máy tính với Robot thông qua cổng Bluetooth, mình sẽ sử dụng thư viện pySerial. Mình có thể tham khảo thêm ở trang Web này: https://pythonhosted.org/pyserial/

Cụ thể như sau:

- Đầu tiên các bạn sẽ kết nối máy tính với moduel Bluetooth HC-05, sau đó mở Device Manager để xem tên port kết nối. Lúc này các bạn sẽ thấy có tận hai Port, ví dụ trong trường hợp của mình là "COM21" và "COM22". Để xác định được cổng nào để kết nối bạn vô **Bluetooth and other Devices** > **More Bluetooth Options** > tab **COM Ports**
- Lúc này bạn sẽ chọn cổng có Direction là **Outgoing**. Ví dụ của mình sẽ là **COM22**


Bluetooth Settings
Options
COM Ports
Hardware
This PC is using the COM (serial) ports listed below. To determine whether you need a COM port, read the documentation that came with your Bluetooth device.
Port
Direction
Name
COM12
Outgoing
'Dev B'
COM16
Incoming
HC-05
COM17
Outgoing
HC-05 'SPP Dev'
COM21
Incoming
REC_PROTOTYPE
COM22
Outgoing
REC_PROTOTYPE 'Dev B'
COM6
Incoming
Add...
Remove

- Sau đó bạn chọn Baudrate thích hợp. Con HC-05 của mình hiện đang giao tiếp với Baudrate 38400.
- Để kết nối, các bạn sử dụng lệnh sau:

```
import serial

ser = serial.Serial("COM22",
baudrate = 38400,
timeout=1)
```

- Lúc này, đèn LED trên HC-05 sẽ chớp tắt chậm, thể hiện đã kết nối thành công

Tuy nhiên, trong những trường hợp mà máy tính không tìm thấy port hoặc con HC-05 chưa được cấp nguồn, thì chương trình của bạn sẽ bị crash. Để giải quyết vấn đề này mình sẽ sử dụng cấu trúc lệnh `try` and `except` của python

```
#Kết nối Robot thông qua thư viện PySerial
while True:
    try:
        print("Connecting...")
        ser = serial.Serial("COM22",
            baudrate = 38400,
            timeout=1)
        break
    #Trong trường hợp không kết nối được
    except serial.SerialException:
        print("No Connection, no serial port found")
        time.sleep(1)

#Thời gian chờ khởi động kết nối
time.sleep(3)
```

Chỉnh lại và ta sẽ có một hàm hoàn chỉnh

```
def serconnect():
    #Chọn Baudrate và Port kết nối
    global ser
    port = "COM22"
    b_rate = 38400
    #Kết nối Robot thông qua thư viện pySerial
    while True:
            try:
                print("Connecting...")
                ser = serial.Serial(port,
                baudrate = b_rate,
                timeout=1, write_timeout = 1)
                break
            #Trong trường hợp không kết nối được
            except serial.SerialException:
                print("No Connection, no serial port found")
                time.sleep(1)
    #Thời gian chờ khởi động kết nối
    time.sleep(3)
```

2 Gửi tín hiệu cho Robot

Để gửi tín hiệu cho Robot, chúng ta sẽ sử dụng hàm `write()` trong thư viện pySerial
Các bạn lưu ý là hàm `write()` chỉ gửi tín hiệu dưới dạng `byte` của Python. Nên để định dạng về `byte` các bạn dùng ký hiệu `b'<ký tự cần gửi>'`

```
ser.write(b'1')
```

Hoặc để thực hiện nhận input dưới dạng hàm thì sử dụng:

```
ser.write(str(input).encode())
```

Bởi vì trong suốt thời gian diễn ra trận đấu, người chơi không được phép can thiệp vào robot tự động, nên việc các bạn xử lý các error và exception rất là quan trọng, tránh các trường hợp bị crash chương trình giữa trận.
Để phòng ngừa trường hợp mất kết nối giữa trận, và tín hiệu không gửi đi được gây crash chương trình, ta tiếp tục sử dụng cấu trúc lệnh `try` and `except`

Về chi tiết các exception, bạn có thể tham khảo trên trang chủ pySerial

```
def sendSignal(input):
    global ser
    port = "COM22"
    b_rate = 38400
    while(True):
        """ Xử lý mất kết nối giữa trận """
        try:
            if(ser == None):
                ser = serial.Serial(port,
                baudrate = b_rate,
                timeout=1,
                write_timeout = 1)
                print("Reconnecting")
            #Gửi dữ liệu tới Robot
            ser.write(str(input).encode())
            time.sleep(0.2)
            #Đọc dữ liệu (nếu có)
            data = ser.readline().decode()
            print("Writing Data...")
            print("Data: "+ data)
            break
```

```
        #Giải quyết việc mất kết nối
        except serial.SerialTimeoutException:
            if(not(ser == None)):
                ser.close()
                ser = None
                print("Write time out")
            print("No Connection, can't write")
            time.sleep(1)
        except serial.SerialException:
            if(not(ser == None)):
                ser.close()
                ser = None
                print("Serial disconnected")
            print("No Connection, no serial port found")
            time.sleep(1)
```

Trong một số trường hợp module Bluetooth bị ngắt kết nối giữa chừng, máy tính có thể vẫn sẽ tiếp tục gửi và mất một lúc sau mới nhận được là do các data đó được gửi vào trong vùng lưu trữ tạm thời. Các bạn có thể thay đổi kích cỡ của vùng lưu trữ đó để loại trừ khoảng thời gian bị delay.

3 Tính toán khoảng cách

Để tính toán khoảng cách từ robot đến một vật, ta có thể sử dụng định lý Pythagoras

```
#Sử dụng định lý Pythagoras
def calculateDistance(autoIndex_c, element_c):
        x_auto = data[autoIndex_c]["position"][0]
        y_auto = data[autoIndex_c]["position"][1]
        x_element = element_c["position"][0]
        y_element = element_c["position"][1]
        return sqrt( (x_auto - x_element)**2 +
                (y_auto - y_element)**2 )
```

4 Sắp xếp thứ tự

Đây là hàm sắp xếp các vật thể trên sân theo thứ tự khoảng cách giảm dần so với con auto bot của đội mình, và chỉnh sửa các thông tin liên quan theo thứ tự đã được sắp xếp. Các vật trên sân mà autobot cần phải lựa chọn sẽ có các thông số về tọa độ, điểm. Sau khi sắp xếp các vật dựa trên tọa độ, chúng ta cần phải cập nhật lại điểm cho các vật đó.

```
#Sắp xếp các phần tử dữ liệu với các đặc tính tương ứng
def bubbleSort(array, indexArray_b, objectSocreRanking_b):
        for count in range(len(array)):
                for count_1 in range(len(array) - count - 1):
                        if (array[count + count_1 + 1] < array[
                                #Sắp xếp theo khoảng cách
                                temporary = array[count]
                                array[count] = array[count + co
                                array[count + count_1 + 1] = te

                                #Cập nhập điểm
                                temporary = objectSocreRanking_
                                objectSocreRanking_b[count] = o
                                objectSocreRanking_b[count + co

                                #Cập nhập index
                                temporary = indexArray_b[count]
                                indexArray_b[count] = indexArra
                                indexArray_b[count + count_1 +
```

5 Tính toán và trả về tín hiệu

Chúng ta cần phải xử lí 3 thông số: tọa độ autobot, tọa độ điểm autobot muốn tới và hướng thực tại của autobot
Hướng của autobot đã sẵn là một vector, chúng ta cần tính vector nối 2 điểm tọa độ của autobot và vật mà autobot muốn tới, từ đó có thể so sánh hướng của 2 vector và gửi tín hiệu cho auto bot:

- Xoay cho trùng hướng (nếu 2 vector khác hướng)
- Chạy thẳng tới vật đó (nếu 2 vector trùng hướng)

Lưu ý: Khái niệm hai vector trùng hướng của toán học rất chính xác, và thực tế autobot rất khó đạt được chính xác trạng thái trùng vì khi quay bot có quán tính. Do đó mình nên dùng phép tính cosin giữa 2 vector và chọn vùng sai số, chứ không nên dùng phép tính tỉ lệ để so sánh 2 vector trùng phương.

```
def returnControl(point_1, point_2, vector):
    #Vector từ điểm hai đến điểm 1
    x_axis = point_1[0] - point_2[0]
    y_axis = point_1[1] - point_2[1]

    #Tính tích vô hướng của hai vector
    lengthVector1 = sqrt(x_axis**2 + y_axis**2)
    lengthVector2 = sqrt(vector[0]**2 + vector[1]**2)
    dotProduct = x_axis*vector[0] + y_axis*vector[1]

    #Trong trường hợp không có Object
    if (lengthVector1*lengthVector2 == 0):
        cosOfAngle = 1
        print("vector multiplication = 0")
    #Trong trường hợp có ít nhất 1 object
    else:
        cosOfAngle = dotProduct / (lengthVector1*lengthVector2)
    print("Cos of Angle:" + str(cosOfAngle))
    if (cosOfAngle > -0.9):
        print ("Rotate")
        return 1
    else:
        print ("Go straight")
        return 2
```

III Chuẩn bị các thông số

Trong trường hợp đang xét trên sân, chỉ có 2 vật là object_1 và auto_1, có số điểm (score) lần lượt là 5, 0 (điểm được quy định trong bảng tham khảo). Vì trong tài liệu này, robot sẽ lấy vật có số điểm xếp hạng (rank) thấp nhất, nên vật có điểm xếp hạng (rank) càng thấp (hạng 1), sẽ có điểm số (score) cao nhất, nên mình sẽ xếp hạng cho các điểm của vật thể, điểm cao nhất tương ứng hạng 1. Sau đó mình sẽ xếp hạng khoảng cách, khoảng cách càng gần thì điểm xếp hạng (rank) càng thấp. Bằng cách cộng điểm hạng 2 tiêu chí điểm và khoảng cách mình có thể chọn ra vật cần thiết.

Lưu ý: rank thấp nhất sẽ được qui định là 1, các rank cao hơn sẽ lần lượt là 2,3,4,...

Vì dữ liệu gửi về từ server sẽ bao gồm thêm cả thời gian của trận đấu, nên để gọi phần dữ liệu tọa độ:

```
data = dataRaw["data"]
```

Trong trường hợp đang xét trên sân, chỉ có 2 vật là object_1 và auto_1 có số index trong bảng tra cứu lần lượt là 20 và 0. Để gọi hai phần tử và gán vào một array mình sử dụng:

```
data = [data[0], data[20]]
```

Tiếp theo, mình sẽ tạo một array tương ứng với số điểm của từng vật. Object_1 có số điểm là 5, và xe auto_1 sẽ không có điểm.

```
objectScore = [5, 0]
```

1	Index	Tổ hợp màu	Tên gọi ("name")	Ý nghĩa	Số điểm
2	0	orange-orange	object_1	Hình hộp	5
3	1	red-red	object_2	Hình hộp	5
4	2	green-green	object_3	Hình hộp	5
5	3	blue-blue	object_4	Hình hộp	5
6	4	yellow-yellow	object_5	Hình hộp	5
7	5	purple-purple	object_6	Hình hộp	5
8	6	pink-pink	object_7	Hình hộp	5
9	7	orange-red	object_8	Hình dẹt	10
10	8	orange-green	object_9	Hình dẹt	10
11	9	orange-blue	object_10	Hình dẹt	10
12	10	orange-yellow	object_11	Hình dẹt	10
13	11	orange-purple	object_12	Hình dĩa	15
14	12	orange-pink	object_13	Hình dĩa	15
15	13	red-green	object_14	Hình dĩa	15
16	14	red-blue	object_15	Hình máy bay giấy	15
17	15	red-yellow	object_16	Hình máy bay giấy	20
18	16	red-purple	object_17	Hình cái bàn	20
19	17	red-pink	object_18	Hình cái bàn	25
20	18	green-blue	manual_1	Xe Manual đội 1	-
21	19	green-yellow	manual_2	Xe Manual đội 2	-
22	20	green-purple	auto_1	Xe Auto đội 1	-
23	21	green-pink	auto_2	Xe Auto đội 1	-
24	22	blue-yellow	object_19	Hình phi tiêu	25

Sau đó, mình bắt đầu tạo một array thiết lập hạng:

```
objectScoreRanking = [1, 2]
```

Gán index cho robot tự động, để chương trình biết robot đang ở vị trí thứ bao nhiêu trong array data

```
autoIndex = 1
```

Set up tọa độ storage zone, với x, y tối đa là 300x300. Storage sẽ nằm ở khu vực trung tâm ở một cạnh.

```
storageZone = [150, 20]
```

```
global data
global ser

data = [data[0], data[20]]

#Khởi tạo
objectScore = [5, 0] #Tương ứng object_1 và auto_1
objectScoreRanking = [1, 2]
autoIndex = 1
storageZone = [150, 20]

distanceArray = []
indexArray = []
scoreArray = []
```

Để minh họa dễ hiểu hơn, ta có thể làm với 5 vật thể: object_1, object_2, object_8, object_13 và auto_1:

```
'''
Object_13 sẽ có rank thấp nhất vì có điểm (score) cao nhất,
Object_1 và Object_2 sẽ có rank bằng nhau vì có số điểm bằng
nhau
'''
data = [data[0], data[1], data[7], data[12], data[20]]

#Khởi tạo
objectScore = [5, 5, 10, 15, 0]
objectScoreRanking = [3, 3, 2, 1, 4]
autoIndex = 4
storageZone = [150, 20]

distanceArray = []
indexArray = []
scoreArray = []
```

IV Kết hợp hàm và phát triển chiến thuật

Tính toán khoảng cách của autobot tới tất cả các vật trên sân và lưu tất cả các thông số tính được vào 1 array:

```
#Tính toán khoảng cách và đưa vào array
for count in range(len(data)):
        distanceArray.append(
                calculateDistance(autoIndex, data[count]))
        indexArray.append(count)
```

Sắp xếp dựa trên array mới tạo và update các thông số liên quan bao gồm điểm và index

```
bubbleSort(distanceArray, indexArray, objectScoreRanking)
```

Chấm điểm cho các vật thể trên sân, dựa trên điểm mình cho mình có thể lựa chọn vật thể phù hợp nhất, ở đây mình cho vật thể phù hợp nhất có điểm thấp nhất. Mình sẽ lựa chọn vật thể đó và lưu vô một biến

Trong trường hợp `distanceArray[count] == 0`, thì đó là khoảng cách từ con bot tới chính con bot. Nên tạm gán xếp hạng 1000000000, để tránh trường hợp nó tự đi tới chính nó

```
#Tính toán điểm các object trên sân
for count in range(len(data)):
        #kiếm index của auto_1
        if (distanceArray[count] != 0):
                scoreArray.append(
                objectScoreRanking[count] + count)
        else:
                scoreArray.append(100000000)

#xem xét lựa chọn tốt nhất
#(trong trường hợp này là ít điểm xếp hạng nhất)
selectionScore = min(scoreArray)
selectionIndex = 0

#Lấy index của lựa chọn tốt nhất trong array
for count in range(len(scoreArray)):
        if selectionScore == scoreArray[count]:
                selectionIndex = indexArray[count]
```

Sau đó mình sẽ tính toán xem là auto bot của mình đã tới sát vật đó chưa, mình sẽ chuẩn bị một biến là previousDistance (khoảng cách autobot-vật được chọn ở frame hình trước) và một biến là currentDistance (khoảng cách autobot-vật được chọn ở frame hình đang xử lí). Nếu 2 biến này xấp xỉ nhau trong khoảng cho phép tức là 2 autobot đã tới được vật đang cần tiến đến, lúc này autobot sẽ giữ vật và quay về storage zone

```
currentDistance =
calculateDistance(autoIndex, data[selectionIndex])

print ("Distance to the selected object: " +
str(currentDistance))

if (abs(currentDistance - previousDistance) > 5):
    #Hướng về Object đã được chọn
    signal = returnControl(data[autoIndex]["position"],
    data[selectionIndex]["position"] ,
    data[autoIndex]["dimension"])
    print (str(signal) + " To selected object")

elif(currentDistance < 95
and abs(currentDistance - previousDistance) <= 5):

    #Hướng về khu vực Storage Zone
    signal = returnControl(data[autoIndex]["position"],
    storageZone ,data[autoIndex]["dimension"])
    print (str(signal) + " To storage zone")

previousDistance = currentDistance
```

Bước cuối cùng là gửi tín hiệu về cho robot:

```
#Gửi tín hiệu cho Robot
sendSignal(signal)
```

V Bỏ tất cả vào một hàm "ảo" để sử dụng trong script lấy data

Vì mình phải kết hợp script xử lí data và script lấy data từ server về nên mình sẽ bỏ hết tất cả những bước tính toán trên vô một hàm, ở đây mình đặt tên là main(). Và file script chứ phần xử lý là dataProcessing.py

Ở script lấy data, cụ thể là file AutobotSample.py các bạn import script xử lí và sử dụng.

VI Những vấn đề có thể nảy sinh

Vấn đề đầu tiên là quán tính của xe. Khi xe chạy với tốc độ nhanh thì quán tính sẽ rất lớn, nên việc chuyển đổi giữa quẹo và đi thẳng nhiều lúc sẽ không chính xác. Một vấn đề nữa, tốc độ của hai động cơ thường sẽ không đều nhau, lúc đi thẳng thì nó lại lệch sang trái hoặc phải, gây khó khăn cho việc lập trình. Một cách giải quyết là dùng encoder, nó sẽ phức tạp hơn nhưng bù lại việc điều khiển xe sẽ hiệu suất và chính xác hơn rất nhiều.

Thời gian delay giữa việc nhận tín hiệu từ server đến lúc gửi cho robot cũng cần phải quan tâm. Các lệnh time.sleep() nên hạn chế sử dụng nhiều, tránh các tình trạng bị độ trễ cao

Các bạn cũng có thể sử dụng hoặc thêm các loại sensor dò đường khác ví dụ như Ultrasonic, lidar hay camera để điều khiển robot một cách chính xác hơn

Thank you

Chúc các bạn một cuộc thi thật hào hứng và tràn đầy niềm vui!