Seguidor de Linha (Imersão)

Ambos os códigos usam o mesmo princípio

1. Booleano

```
if cancel:
    return

global run_generator, runSmall
```

A condicional “if cancel:” no código é usada para verificar se a variável chamada cancel é uma condição verdadeira. Se for, a função retornará imediatamente e interromperá a execução. Isso pode ser usado como uma forma de interromper a execução da função, por exemplo, se a função estiver sendo executada em um loop e você quiser interrompê-la prematuramente. A variável (cancel) precisaria ser definida como True em algum lugar fora da função para que a função parasse mais cedo.

Exemplo:

cancel = True

return

Run_generator, runSmall

print (GYROSTRAIGHT INTERROMPIDO)

2. Booleano

```
if generator == None:
    run_generator = False
```

A linha if generator == None: no código fornecido é usada para verificar se a variável generator é None. Se for, a variável run_generator é definida como False.

Isso é usado posteriormente no código quando a função verifica o valor de run_generator para determinar se o código deve ser executado dentro do bloco if run_generator:. Se run_generator for False, o código dentro do bloco não será executado.

Portanto, o propósito dessa linha é desativar a execução do código dentro do bloco if run_generator: quando a variável generator é None. Isso pode ser útil se a variável generator não for passada como argumento ao chamar a função ou se for definida explicitamente como None dentro da função.

3. Definições e Cálculos

```
#Definir a velocidade em que o robô começa
speed = startspeed
#Redefinir os valores PID para eliminar bugs
change = 0
old_change = 0
integral = 0
#Redefinir a distância percorrida do robô para eliminar bugs

#Especifica o sensor de cor
colorsensor = ColorSensor('C', 'D' )
#Obtenha os graus dos motores girados antes que o robô se mova, permitindo o cálculo da distância sem reiniciar os m
loop = True
#Retroceder não é suportado em nosso robô devido aos motores estarem na frente dos sensores de cores e o programa nã
if distance < 0:
    print('ERR: distance < 0')
    distance = abs(distance)
#Calcule os valores alvo para os motores girarem
finalDistance = (distance / 17.6) * 360
#Calcule após que distância o robô tem para atingir a velocidade máxima
accelerateDistance = finalDistance * addspeed
deccelerateDistance = finalDistance * (1 - brakeStart)

invert = 1

#Cálculo do fator de direção, dependendo de qual lado da linha estamos
if side == "left":
    invert = 1
elif side == "right":
    invert = -1
```

“””

change = 0

old_change = 0

integral = 0

“””

Estas linhas definem e inicializam três variáveis relacionadas a um controlador PID (Proporcional-Integral-Derivativo), que é usado para regular o movimento do robô. Estas variáveis são usadas para calcular o erro entre a velocidade desejada e a velocidade real dos motores do robô.

“””

```
if distance < 0:
    print('ERR: distance < 0')
    distance = abs(distance)
```

“””

Este código verifica se a distância a ser percorrida pelo robô é negativa. Se for, o código imprime uma mensagem de erro e define a distância como seu valor absoluto. Isso é provavelmente porque o robô não pode se mover para trás devido à posição dos sensores de cores e motores.

“””

```
invert = 1

if side == "left":
    invert = 1
elif side == "right":
    invert = -1
```

“””

Este código calcula um fator de direção com base no lado da linha em que o robô está. Se o robô estiver do lado esquerdo, o fator é definido como 1, e se estiver do lado direito, o fator é definido como -1.

INFORMAÇÂO:

Fiz esses conteúdos indo do mais básico ao mais avançado então alguns conceitos daqui eu pulei pois expliquei em outras Imersões, fé fé.