GyroTurn (Imersão)

Usando os mesmos princípios da imersão do GyroMove

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_generetor, runSmall = 0

print (GYROTURN 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. Execução e Calibragem

```
if rotate_mode == 0:
    startspeed = abs(startspeed)
    maxspeed = abs(maxspeed)
    endspeed = abs(endspeed)

speed = startspeed

#Definir variáveis padrão
rotatedDistance = 0
steering = 1

accelerateDistance = abs(angle * addspeed)
deccelerateDistance = abs(angle * (1 - brakeStart))

#Calibração do sensor giroscópio
angle = angle * (2400/2443) #Valor experimental baseado em 20 rotações d

#Setting variables based on inputs
loop = True
gyroStartValue = getGyroValue() #Ângulo de guinada usado devido à orient
brakeStartValue = (angle + gyroStartValue) * brakeStart

#Inversão do valor de direção para girar no sentido anti-horário
if angle < 0:
    steering = -1
```

```
while loop:
    if cancel:
        break

    if run_generator: #Executar execução paralela de código
        next(generator)

    oldRotatedDistance = rotatedDistance
    rotatedDistance = getGyroValue() #Yaw angle used due to orientation of the self.hub. This migh
    speed = speedCalculation(speed, startspeed, maxspeed, endspeed, accelerateDistance, deccelerat


    #Checking for variants
    #Both Motors turn, robot moves on the spot
    if rotate_mode == 0:
        self.movement_motors.start_tank_at_power(int(speed) * steering, -int(speed) * steering)
    #Only outer motor turns, robot has a wide turning radius

    elif rotate_mode == 1:

        if angle * speed > 0:
            self.leftMotor.start_at_power(- int(speed))
        else:
            self.rightMotor.start_at_power(+ int(speed))

    if stopMethod != None:
        if stopMethod.loop():
            loop = False
            break
    elif abs(angle) <= abs(rotatedDistance - gyroStartValue):
            loop = False
            break
```

- Primeiro verifica o valor da variável rotate_mode.
- Se for 0, ele converte as velocidades inicial, máxima e final em valores absolutos.
- Ele define a velocidade inicial para a velocidade inicial.
- Ele inicializa diversas variáveis, incluindo

❑ RotatedDistance (que monitora a distância girada)
❑ Steering (que determina a direção da rotação)
❑ AccelerateDistance (que determina até que ponto o robô deve acelerar)
❑ DeccelerateDistance (que determina até que ponto o robô deve desacelerar).).

- Ele calibra o valor do sensor giroscópio multiplicando o ângulo por um fator constante (2400/2443).
- Ele define vários sinalizadores e variáveis com base na entrada, incluindo loop (que determina se o loop while deve continuar), gyroStartValue (que armazena o valor inicial do giroscópio) e BrakeStartValue (que armazena o valor no qual o robô deve iniciar a frenagem).
- Ele verifica o sentido de rotação e ajusta a variável de direção de acordo.
- Ele entra em um loop while que continua até que o loop seja definido como False. Dentro do loop
- Ele verifica se o sinalizador de cancelamento está definido e, em caso afirmativo, sai do loop.

No geral, este código controla a rotação de um robô ou outro dispositivo com base em um determinado ângulo e velocidade. Ele usa o sensor giroscópio para medir o ângulo atual e ajusta a velocidade e a direção de rotação de acordo.

4. Método de finalizar a execução

```
if stop:
    self.movement_motors.stop()
```

5. Difunde para um próximo código

```
run_generator = True
runSmall = True
```