Projetinho PYTHON 2024

Conteúdo:

- Introdução
- GyroStraight
- GyroTurn
- Seguidor de linha
- ArcRotation = Parábola
- Plano Cartesiano

INTRODUÇÃO

Python é uma linguagem de programação de alto nível, interpretada, de tipagem dinâmica e de fácil leitura. Foi criada por Guido van Rossum e lançada pela primeira vez em 1991. Sua sintaxe simples e expressiva, juntamente com uma vasta biblioteca padrão, tornaram-na popular em uma variedade de domínios, incluindo desenvolvimento web, automação, análise de dados, inteligência artificial, entre outros.

Vantagens do Python:

- Legibilidade de Código: A sintaxe clara e intuitiva do Python facilita a leitura e a manutenção do código, contribuindo para um desenvolvimento mais eficiente.
- Produtividade: Python é conhecido por sua produtividade, permitindo aos desenvolvedores realizarem tarefas em menos linhas de código em comparação com outras linguagens.
- Diversidade de Aplicações: Python é versátil e pode ser utilizado em uma variedade de aplicações, desde desenvolvimento web até aprendizado de máquina e automação.
- Comunidade Ativa: A comunidade Python é grande e ativa, o que significa que há uma abundância de recursos, bibliotecas e suporte disponíveis.
- Multiplataforma: Python é executado em várias plataformas, como Windows, Linux e MacOS, proporcionando portabilidade aos projetos.

GyroStraight

A função GyroStraight permite que o robô avance ou retroceda em linha reta, usando o giroscópio para ajustar a direção. Ela aceita diversos parâmetros para personalizar o comportamento do robô durante a movimentação.

O bloco de código a seguir define a função GyroStraightDrive:

(MYBLOCK)

```
def gyroStraightDrive(self, distance, startspeed, maxspeed, endspeed, addspeed = 0.3,
```

Dentro da função, várias variáveis são declaradas para armazenar os valores das entradas do usuário, bem como valores intermediários usados no cálculo do desvio do robô e da velocidade.

Além disso, a função também verifica se a distância fornecida é válida (ou seja, não é negativa).

O bloco de código principal da função gyroStraightDrive envolve um loop while que é executado até que a distância desejada seja alcançada. Dentro do loop, a função realiza as seguintes etapas:

1. Verifica se a movimentação paralela está habilitada e, em caso afirmativo, executa a próxima iteração do gerador.
2. Calcula a diferença entre a posição atual do giroscópio e a posição desejada.
3. Atualiza os valores do PID com base na diferença calculada e nas condições atuais do robô.
4. Calcula o valor do desvio para ajustar a direção do robô.
5. Atualiza a velocidade do robô com base na aceleração, desaceleração e condições atuais do robô.
6. Verifica se a função de parada fornecida (stopMethod) indica que o robô deve parar e, em caso afirmativo, interrompe o loop.
7. Caso contrário, se a distância desejada tiver sido alcançada, interrompe o loop.
8. Após o término do loop, a função interrompe os motores do robô, se necessário.

A função gyroStraightDrive pode ser usada em várias aplicações, como navegação, control de precisão e manutenção de robôs em ambientes fechados.

GyroTurn

A função GyroTurn é usada para fazer o robô girar o comprimento de um ângulo específico ou para o robô girar até sentir uma linha. O robô pode acelerar e desacelerar durante essa função. Além disso, a função possui calibrações de GyroSensor baseadas na experiência experimental do autor.

Vamos analisar o código passo a passo:

1. Verifica se o usuário cancelou a operação. Se sim, retorna imediatamente.
2. Define variáveis globais para executar o gerador em paralelo com a condução do robô.
3. Verifica o valor de rotate_mode. Se for 0, ele torna as velocidades de início, máxima e final absolutas.
4. Define variáveis padrão, como speed, rotatedDistance, e steering.
5. Calcula as distâncias de aceleração e desaceleração, bem como os valores de parada para a rotação.
6. Corrige a orientação do ângulo de rotação para girar no sentido anti-horário.
7. Cria um loop para girar o robô.

a. Executa o gerador em paralelo, se fornecido.

b. Calcula a nova velocidade com base nas velocidades iniciais, máximas e finais, e nas distâncias de aceleração e desaceleração.

c. Verifica se o stopMethod foi fornecido. Se for, usa o stopMethod para parar a rotação. Se não, usa a diferença entre a distância girada e a distância de partida para parar a rotação.

d. Se o stopMethod retornar True, a rotação será interrompida.

8. Se o usuário escolheu parar os motores após dirigir, os motores são interrompidos.
9. As variáveis globais são definidas como verdadeiras, permitindo que o gerador seja executado em paralelo.
10. A função retorna.

Observe que essa função usa uma combinação de motores internos e externos para controlar a direção e a velocidade durante a rotação. Além disso, a função usa o ângulo de guinada do GyroSensor para determinar a direção e a velocidade.

Essa função pode ser usada em diferentes contextos, como em uma linha de bateria ou em uma tarefa de rolagem de bola. O parâmetro rotate_mode permite que o robô gire de diferentes maneiras, como girar no mesmo lugar ou girar em um ângulo largo. O parâmetro stopMethod permite que o robô seja parado usando diferentes métodos, como stopDistance ou algum outro critério personalizado.

Seguidor de linha

A função **LineFollower** é usada para fazer o robô seguir uma linha, girando até que a distância especificada seja alcançada ou o outro sensor do robô detecte uma linha. O robô pode acelerar e desacelerar durante essa função. Além disso, a função possui calibrações de Sensor de Cor baseadas na experiência experimental do autor.

Vamos analisar o código passo a passo:

1. Verifica se o usuário cancelou a operação. Se sim, retorna imediatamente.
2. Define variáveis globais para executar o gerador em paralelo com a condução do robô.
3. Verifica se o usuário escolheu seguir uma linha na direção "left" ou "right".
4. Define variáveis padrão, como **speed**, **change**, **old_change**, **integral**, e **invert**.
5. Calcula os valores alvo para os motores girarem, a distância após a qual o robô precisa atingir a velocidade máxima, e a distância após a qual o robô precisa desacelerar.
6. Redefine a distância percorrida do robô para eliminar bugs.
7. Cria um loop para fazer o robô seguir a linha.

a. Executa o gerador em paralelo, se fornecido.

b. Calcula a nova velocidade com base nas velocidades iniciais, máximas e finais, e nas distâncias de aceleração e desaceleração.

c. Verifica se o stopMethod foi fornecido. Se for, usa o stopMethod para parar a condução. Se não, usa a diferença entre a distância percorrida e a distância de partida para parar a condução.

d. Se o stopMethod retornar True, a condução será interrompida.

8. Se o usuário escolheu parar os motores após dirigir, os motores são interrompidos.
9. As variáveis globais são definidas como verdadeiras, permitindo que o gerador seja executado em paralelo.
10. A função retorna.

Observe que essa função usa uma combinação de motores internos e externos para controlar a direção e a velocidade durante a condução. Além disso, a função usa o Sensor de Cor do robô para determinar a direção e a velocidade.

Essa função pode ser usada em diferentes contextos, como em uma linha de bateria ou em uma tarefa de rolagem de bola. O parâmetro **Side** permite que o robô siga a linha na direção "left" ou "right". O parâmetro **StopMethod** permite que o robô seja parado usando diferentes métodos, como stopDistance ou algum outro critério personalizado.

O PID (Proporcional, Integral e Derivativo) é usado para calcular o fator de direção do robô. Isso ajuda a garantir que o robô siga a linha corretamente e que ele se adapte a eventuais variações na luminosidade da linha. O PIDCalculationLight é usado para calcular os valores PID a partir das informações fornecidas pelo Sensor de Cor.]

A função **SpeedCalculation** é usada para calcular a velocidade atual do robô com base nas velocidades iniciais, máximas e finais, e nas distâncias de aceleração e desaceleração. A função retorna à velocidade atual do robô.

Ela usa um conjunto de condições para determinar se o robô deve acelerar, desacelerar ou permanecer na velocidade máxima. A velocidade atual do robô é calculada com base nas equações lineares das distâncias de aceleração e desaceleração, e na velocidade máxima do robô.

ArcRotation

O código **arc_rotation** é uma função que permite que o robô siga uma trajetória circular com um determinado raio e ângulo. A função usa o módulo **math** para calcular a distância percorrida e a velocidade do robô durante a rotação.

A função **arc_rotation** usa o módulo **math** para calcular a distância percorrida e a velocidade do robô durante a rotação. Ela usa um loop para fazer o robô seguir a trajetória circular, com base no raio e no ângulo fornecidos.

A função calcula a distância percorrida em um círculo completo, a distância percorrida até atingir a velocidade máxima, e a distância percorrida até desacelerar. Ela também calcula a velocidade do robô durante a rotação, com base no raio e na velocidade máxima fornecidos.

A função usa o módulo **math** para calcular a distância percorrida por volta, com base no raio e no número de voltas que o robô deve dar. Ela também usa o módulo **math** para calcular a velocidade do robô durante a rotação, com base no raio e na velocidade máxima fornecidos.

A função usa o módulo **math** para calcular a distância percorrida pelo robô, com base no número de voltas que o robô deve dar e a distância percorrida por volta. Ela também usa o módulo **math** para calcular o valor alvo do robô como a borda das linhas pretas e brancas.

A função usa o módulo **math** para calcular o fator de direção usando PID, definindo um novo valor I. Ela também usa o módulo **math** para calcular a velocidade atual do robô, com base nas velocidades iniciais, máximas e finais, e nas distâncias de aceleração e desaceleração.

A função usa o módulo **math** para atualizar os valores PID e para parar os motores do robô, se o usuário escolheu parar os motores após dirigir.

Em resumo, a função **arc_rotation** usa o módulo **math** para calcular a distância percorrida e a velocidade do robô durante a rotação, com base no raio e no ângulo fornecidos. Ela também usa o módulo **math** para calcular o valor alvo do robô como a borda das linhas

pretas e brancas, e para calcular o fator de direção usando PID. A função usa um loop para fazer o robô seguir a trajetória circular, com base no raio e no ângulo fornecidos.

Plano Cartesiano (Projeto)

```
Click here to ask Blackbox to help you code faster
from spike import PrimeHub, LightMatrix, Button, ForceSensor, MotionSensor, Speaker, Co
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import greater_than, greater_than_or_equal_to, less_than, less_than

# Inicializar o hub
hub = PrimeHub()

# Inicializar o sensor de cor
color_sensor = ColorSensor('C')

# Inicializar o sensor de distância
distance_sensor = DistanceSensor('D')

# Inicializar o motor esquerdo
left_motor = Motor('A')

# Inicializar o motor direito
right_motor = Motor('B')

# Função para mover o robô para uma posição no plano cartesiano
Comment Code
def move_to(x, y):
    # Calcular a distância para mover o robô
    distance = math.sqrt(x**2 + y**2)

    # Calcular o ângulo para mover o robô
    angle = math.atan2(y, x)

    # Mover o robô para a posição desejada
    left_motor.run_angle(distance * -1, angle * 180 / math.pi)
    right_motor.run_angle(distance, angle * 180 / math.pi)
```