UNIVERSIDADE LUSÓFONA

Pick and Place com Webots

Nome: Mauro Carvalho – 22402165

Professor: Plínio Lopez

Data: 09/06/2025

Repositório GitHub: [CMauro1910/pick-and-place-competition]

Pick and Place

1. Introdução 4
2. Metodologia 4
3. Explicação do Código 4
Inicialização do Robô e Configuração Básica 5
Configuração das Rodas 5
Configuração dos Motores do Braço 5
Configuração dos Sensores de Posição do Braço 5
Configuração da Garra (Gripper) 6
Movimento inicial para Frente 6
Posiçao Inicial do Braço 7
Controle Proporcional da Junta 3 do Braço 7
Fechamento da Garra 8
Levantamento do Braço 8
Rotação do Robô 9
Movimento para Frente (Novamente) 9
Descarga do Objeto 9
Abertura da Garra 10
Retorno do Braço à Posição Inicial 10
4. Resultados 11
5. Conclusão 12

1. Introdução

Este trabalho descreve o desenvolvimento de um braço articulado automatizado para o robô YouBot no simulador Webots, com o objetivo da realização de uma tarefa de coleta e entrega de um objeto (pick and place). O objetivo é comparar controle em malha aberta com malha fechada, realizando modificações no código-base fornecido pelo professor.

2. Metodologia

Foram seguidos os seguintes passos dados pelo professor:
- Análise do código original;
- Substituição do controle em malha aberta (temporização fixa) por malha fechada com feedback do sensor;
- Execução de testes para comparar desempenho;
- Coleta de dados de tempo de execução e taxa de sucesso consoante a tarefa.

3. Explicação do Código

O código foi modificado na parte que controla o motor arm4, substituindo o temporizador por um laço de controle proporcional baseado na diferença entre a posição alvo e a posição lida do sensor:

```
python
armMotors[3].setPosition(float('inf'))
target = -1.5
kp = 3.0
tolerance = 0.03
while robot.step(timestep) != -1:
    erro = target - armPositionSensors[3].getValue()
    if abs(erro) < tolerance:
        break
    armMotors[3].setVelocity(max(min(kp * erro, 1.57), -1.57))
armMotors[3].setVelocity(0.0)
```

Esse controle substitui um simples robot.step(500 * timestep), garantindo precisão e adaptabilidade na sua execução.

Inicialização do Robô e a sua Configuração Básica

```
robot = Robot()

timestep = int(robot.getBasicTimeStep())
```

Função: Cria uma instância do robô e obtém o timestep básico da simulação.

Configuração das Rodas

```
wheels = [robot.getDevice(f"wheel{i+1}") for i in range(4)]

for wheel in wheels:

    wheel.setPosition(float('+inf'))
```

Função:

Obtém referências para os 4 motores das rodas, configura cada roda para trabalhar em modo de velocidade (position = +inf).

Configuração dos Motores do Braço

```
armMotors = [robot.getDevice(f"arm{i+1}") for i in range(5)]

armMotors[0].setVelocity(0.2)

armMotors[1].setVelocity(0.5)

armMotors[2].setVelocity(0.5)

armMotors[3].setVelocity(0.7)
```

Funçao:

Obtém referências para os 5 motores do braço robótico e define diferentes velocidades para cada junta do braço do robô.

Configuração dos Sensores de Posição do Braço

```
armPositionSensors = [robot.getDevice(f"arm{i+1}sensor") for i in range(5)]

for sensor in armPositionSensors:
```

sensor.enable(timestep)

Funçao:

- Habilita os sensores de posição para cada junta do braço
- Configura a frequência de atualização dos sensores

Configuração da Garra (Gripper)

finger1 = robot.getDevice("finger1")

finger2 = robot.getDevice("finger2")

finger1.setVelocity(0.05)

finger2.setVelocity(0.05)

fingerMinPosition = finger1.getMinPosition()

fingerMaxPosition = finger1.getMaxPosition()

Funçao:

- **Obtém referências para os dois dedos da garra**
- **Define velocidades de movimento para os dedos**
- **Armazena as posições mínima e máxima dos dedos**

Movimento inicial para Frente

omega = 12

distancia = 2.912

angulo = distancia / 0.05

tempo = angulo / omega

for wheel in wheels:

wheel.setVelocity(omega)

robot.step(int(tempo * 1000) + 16)

for wheel in wheels:

wheel.setVelocity(0.0)

Fórmula: tempo = (distância / raio_da_roda) / velocidade_angular

Função:

- Calcula o tempo necessário para percorrer 2.912m a 12 rad/s
- Move todas as rodas para frente na velocidade especificada definida
- Aguarda o tempo calculado e depois para as rodas

Posiçao Inicial do Braço

armMotors[0].setPosition(0.05)

armMotors[1].setPosition(-0.55)

armMotors[2].setPosition(-0.95)

armMotors[3].setPosition(-1.35)

finger1.setPosition(fingerMaxPosition)

finger2.setPosition(fingerMaxPosition)

Função:

- Posiciona cada junta do braço em posições específicas
- Abre a garra completamente (posição máxima)

Controle Proporcional da Junta 3 do Braço

armMotors[3].setPosition(float('inf'))

target = -1.5

kp = 3.0

tolerance = 0.03

while robot.step(timestep) != -1:

```
    erro = target - armPositionSensors[3].getValue()
    if abs(erro) < tolerance:
        break
    armMotors[3].setVelocity(max(min(kp * erro, 1.57), -1.57))
armMotors[3].setVelocity(0.0)
```

Função:

- **Implementa um controlador proporcional (P) para posicionar a junta 3**
- **Usa feedback do sensor para ajustar a velocidade até atingir a posição alvo**
- **Limita a velocidade máxima a ±1.57 rad/s**

Fechamento da Garra

finger1.setPosition(0.013)

finger2.setPosition(0.013)

robot.step(50 * timestep)

Função:

- Fecha a garra para segurar o objeto
- Aguarda 50 timesteps para completar o movimento

Levantamento do Braço

armMotors[1].setPosition(0)

robot.step(200 * timestep)

Função:

- **Levanta o braço para posição vertical**
- **Aguarda 200 timesteps para completar o movimento**

Rotação do Robô

omega_rot = 7

tempo_rot = 27.5 / omega_rot

wheels[0].setVelocity(omega_rot)

wheels[1].setVelocity(-omega_rot)

wheels[2].setVelocity(omega_rot)

wheels[3].setVelocity(-omega_rot)

robot.step(int(tempo_rot * 1000))

for wheel in wheels:

wheel.setVelocity(0.0)

Função:

- **Gira o robô aproximadamente 180 graus (27.5 radianos)**
- **Configura rodas opostas com velocidades opostas para rotação**
- **Para após o tempo calculado**

Movimento para Frente (Novamente)

omega = 2.5

distancia = 1.8

tempo = (distancia / 0.05) / omega

for wheel in wheels:

wheel.setVelocity(omega)

robot.step(int(tempo * 1000))

Função:

- Move o robô para frente por 1.8m a 2.5 rad/s

Descarga do Objeto

armMotors[3].setPosition(0)

armMotors[2].setPosition(-0.3)

robot.step(200 * timestep)

armMotors[1].setPosition(-1.0)

robot.step(200 * timestep)

armMotors[3].setPosition(-1.0)

robot.step(200 * timestep)

armMotors[2].setPosition(-0.4)

robot.step(50 * timestep)

Função:

- Sequência de movimentos do braço para posicionar o objeto no local da descarga
- Movimenta várias juntas de forma coordenada

Abertura da Garra

finger1.setPosition(fingerMaxPosition)

finger2.setPosition(fingerMaxPosition)

robot.step(50 * timestep)

função:

- **Abre a garra para liberar o objeto**
- **Aguarda 50 timesteps para completar o movimento**

Retorno do Braço à Posição Inicial

armMotors[1].setPosition(0)

robot.step(200 * timestep)

Funçao: Levanta o braço para posição vertical novamente

4. Resultados

Tabelas de Execuções:

Velocidade angular (rad/seg)	7	8	9	10	11	12	13	14
Tempo medio de execução (min:seg)	01:01	01:00:14	00:59:61	00:58:68	00:58:7	00:57:45	00:58:56	00:57:95
Percentagem de sucesso	100%	100	100	75	100	100	100%	100%

Estatísticas:

- Tempo médio: 59.6s

Velocidade angular (rad/seg)	0.3	0.4	0.5	0.6	0.7
Tempo medio de execução (min:seg)	00:58:89	00:57:12	00:56:30	00:55:90	00:55:47
Percentagem de sucesso	75%	100%	100%	100%	100%

A tabela mostra que:

- Quanto **maior a velocidade** (de 0.3 até 0.7 rad/seg), **menor o tempo** de execução (de 58.89s até 55.47s).
- Velocidades **acima de 0.4 rad/seg** garantem **100% de sucesso**.
- A velocidade **mais lenta (0.3 rad/seg)** teve apenas **75% de sucesso**, indicando que é pouco confiável.

Melhor opção: 0.7 rad/seg (mais rápido e 100% eficiente).

Velocidade angular (rad/seg)	0.7	0.8	0.9	1	1.1	1.2
Tempo medio de execução (min:seg)	00:56:83	00:56:54	00:56:11	00:56:32	00:56:25	00:56:30
Percentagem de sucesso	100%	75%	100%	100%	100%	100%

Para a tabela acima, qualquer velocidade entre 0.7 e 1.2 rad/seg funciona bem, menos 0.8 rad/seg (menos confiável). O tempo praticamente não varia.

5.Conclusão

A alteração do controle do motor do braço de malha aberta para malha fechada trouxe maior precisão e confiabilidade na execução da tarefa. O sistema tornou-se mais robusto a variações e o robô obteve 100% de sucesso nas tentativas. A prática demonstrou os benefícios do uso de feedback para controle fino em sistemas robóticos.