UNIVERSIDADE LUSÓFONA

Pit Scape

Nome: Mauro Carvalho – 22402165

Professor: Plínio Lopez

Data: 09/06/2025

Repositório GitHub: https://github.com/CMauro1910/Pit-Scape---Mauro-Carvalho.git

1. Introdução

Este trabalho avalia as limitações do controle em malha fechada no simulador Webots, focando na tarefa de um robô tentar sair de um buraco. O objetivo é mudar a frequência de atualização do controle (timeInterval) para otimizar o desempenho do robô, comparando diferentes valores e os seus impactos no sistema.

2. Metodologia:

Para a execuçao deste trabalho seguiu-se passos simples, mudando apenas valores simples, mas que impactavao de forma diferente no comportamento do robô.

Além de se seguir os seguintes dados pelo professor:

-Análise do código original;

- Execução de testes para comparar desempenho;
- Coleta de dados de tempo de execução e taxa de sucesso consoante a tarefa

Configuração Inicial:

- Fork do repositório original (pit-escape-competition).
- Adaptação do código participant.py para incluir controle proporcional.

Parâmetros Testados:

- timeInterval: 1.4s, 1.45s, 1.5s, 1.55s, 1.6s (3 execuções cada).
- Métricas: **Desempenho (%)** e **Frequência do Controlador (Hz)**.

Coleta de Dados:

- Média de desempenho calculada para cada configuração.

3. Explicação do Código

Como Sitado anteriormente, o código foi modifido apenas na variável timeInterval (definida na linha 22):

timeInterval = 1.5

Inicialização do Robô

from controller import Robot

robot = Robot()

timestep = int(robot.getBasicTimeStep())

Função:

Importa a biblioteca do Webots e cria o objeto do robô.

timestep define o intervalo de tempo entre atualizações do controle (em ms)

Valor padrão é normalmente 32ms

2. Configuração do Motor Principal

maxSpeed = 8.72

pitchMotor = robot.getDevice("body pitch motor")

pitchMotor.setPosition(float('inf'))

pitchMotor.setVelocity(0.0)

- **Função**: Prepara o motor que controla o movimento de subida/descida do robô.
    - setPosition(float('inf')) coloca o motor em modo de controle de velocidade
    - Velocidade inicial é zero

3. Configuração do Giroscópio

```
gyro = robot.getDevice("body gyro")
gyro.enable(timestep)
```

- **Função**: Ativa o sensor que mede a rotação do robô.
    - O giroscópio ajuda a detectar se o robô está inclinando para frente/trás
    - enable(timestep) faz o sensor atualizar suas leituras a cada passo de simulação

4. Lógica Principal de Controle

```
timeInterval = 1.5
pitchMotor.setVelocity(maxSpeed)
forward = True
lastTime = 0
values_old = gyro.getValues()
```

- **Função**: Inicializa as variáveis para o controle periódico.
    - timeInterval: tempo entre verificações de direção (1.5 segundos)
    - Começa movendo para frente na velocidade máxima

5. Loop Principal

```
while robot.step(timestep) != -1:
    now = robot.getTime()
    if now - lastTime > timeInterval:
        values_new = gyro.getValues()
        delta = values_new[1] - values_old[1]
```

```
if delta > 0.1:
    pitchMotor.setVelocity(maxSpeed)
elif delta < -0.1:
    pitchMotor.setVelocity(-maxSpeed)
else:
    pitchMotor.setVelocity(0.0)

lastTime = now
values_old = gyro.getValues()
```

Função: Controla o movimento do robô com base na aceleração medida.

A cada timeInterval (1.5s), verifica a variação da velocidade angular (delta)

Se delta > 0.1 (aceleração positiva): move para frente

Se delta < -0.1 (aceleração negativa): move para trás

Caso contrário: para o motor

Atualiza o tempo da última verificação e os valores do giroscópio

Funcionamento Geral:

O robô inicia a sua trajetória para frente na velocidade máxima. A cada 1,5 segundos, o robô verifica:

A subida (acelerando) → continua para a frente

A descida (desacelerando) → muda e vai para trás

A subida e descida → para (provável saída do buraco)

No final das contas, é como se o robô ficasse "a tentar andar para a frente e para a trás" até conseguir sair do buraco sozinho. Quando o robô para de se mexer, é sinal que conseguiu sair.

Resultados

	timeInterval =1.4s	timeInterval=1.45s	timeInterval=1.5s	timeInterval=1.55s	timeInterval=1.6s
Média do desempenho (%)	60,11%	**58,88%**.	43,71%	37,07%	73,82%
Frequência do controlador	0.714 Hz	0.690 Hz	0.667 Hz	0.645 Hz	0.625 Hz

A tabela acima mostra a média de desempenho e a frequência que foi obtida através da média dos valores retirados em três testes.

Análise:

A configuração de 1.6s foi a que deu mais certo (73,8% de sucesso)
Quanto maior o intervalo (1.4s → 1.6s), melhor o resultado (exceto um caso)
Usar 1.6 segundos entre ajustes deu os melhores resultados. Testar tempos um pouco maiores pode ajudar a melhorar ainda mais

Conclusão

O controle em malha fechada mostrou-se eficaz, e a otimização da frequência de atualização é crucial para o desempenho. Para este robô e tarefa específicos, um intervalo de 1.6 segundos entre atualizações de controle provou ser o mais adequado, equilibrando precisão e eficiência energética.