DOCUMENTACAO PROJETO DE ROBOTICA E INTELIGENCIA ARTIFICIAL

INTEGRANTES:
LUCAS ANTUNES SAMPAIO – R.A: 24.122.056-5

ROMULO CARNEIRO DE OLIVEIRA CANAVESSO - R.A: 24.122.093-8

1. DESCRICAO DO PROJETO

O projeto consiste em realizar uma simulação no WeBots utilizando o robô e-puck com o objetivo de:

- O robô deve navegar pelo cenário com diversas caixas.
- O robô deve encontrar a caixa mais leve e possível de ser empurrada.
- O robô deve empurrar a caixa.
- Após empurrar a caixa, o robô deve encerrar sua navegação e começar a girar em seu próprio eixo

2. METODOLOGIA

Para alcançar os objetivos propostos, foi decidido que o melhor método foi de utilizar as posições das caixas iniciais e salva-las em uma matriz no robô, assim, o robô saberia quais são todas as posições originais das caixas, então caso ele deslocasse uma caixa (no caso, a caixa mais leve) de sua posição original, ele para e começa a girar em seu eixo, o código “PosicaoCaixas.c”, disponibilizado pelo professor, foi empregado para acessar as posições de todas as caixas e envia-las para o robô.

3. TRECHOS DOS CODIGOS

Nesse trecho, definimos como:

TIME_STEP - intervalo de tempo para atualização dos sensores/atuadores.

QtddSensoresProx - o e-puck tem 8 sensores de proximidade.

QtddCaixa - há 18 caixas na simulação.

DELTA - tolerância para considerar que uma caixa foi movida. Usado para comparar posições antes e depois. Um valor muito pequeno, pois até movimentos leves são relevantes.

```
#define TIME_STEP 256
#define QtddSensoresProx 8
#define QtddLeds 10
#define QtddCaixa 18
// Tivemos que mudar o DELTA
#define DELTA 0.002
```

Nesse trecho temos:

Inicialização do robô e de utilitários para realizar a matriz das posições das caixas.

Configuração dos motores para movimento contínuo (INFINITY) e velocidade zero inicialmente.

Leitura e ativação dos 8 sensores de proximidade ps0 até ps7.

```
int main(int argc, char **argv) {
  int i, j;
  char texto[256];
  double LeituraSensorProx[QtddSensoresProx];
  double AceleradorDireito = 1.0, AceleradorEsquerdo = 1.0;
  bool caixaMovida = false;

  wb_robot_init();

  WbDeviceTag MotorEsquerdo = wb_robot_get_device("left wheel motor");
  WbDeviceTag MotorDireito = wb_robot_get_device("right wheel motor");
  wb_motor_set_position(MotorEsquerdo, INFINITY);
  wb_motor_set_position(MotorDireito, INFINITY);
  wb_motor_set_velocity(MotorEsquerdo, 0);
  wb_motor_set_velocity(MotorDireito, 0);

  WbDeviceTag SensorProx[QtddSensoresProx];
  for (i = 0; i < QtddSensoresProx; i++) {
    char nome[4];
    sprintf(nome, "ps%d", i);
    SensorProx[i] = wb_robot_get_device(nome);
    wb_distance_sensor_enable(SensorProx[i], TIME_STEP);
  }
```

Nesse trecho temos:

Inicialização dos LEDs, mas no código só o led0 é usado (pisca alternadamente).

```
WbDeviceTag Leds[QtddLeds];
Leds[0] = wb_robot_get_device("led0");
wb_led_set(Leds[0], -1);
```

Nesse trecho temos:

Declaração de arrays: “caixa[]” armazena referências para os objetos "CAIXA00", "CAIXA01", ..., "CAIXA17" do simulador Webots, “posicoesIniciais[][]” armazena a posição (x, y, z) de cada caixa no início da simulação.

Loop for que itera por todas as caixas.

“wb_supervisor_node_get_from_def(nomeCaixa)”, acessa a referência da caixa no ambiente do Webots usando seu nome (como "CAIXA00", "CAIXA01", etc.).

Usa o Supervisor API para permitir que o robô acesse a posição de qualquer objeto da simulação.

“wb_supervisor_node_get_position(caixa[i])”, obtém a posição atual da caixa no espaço 3D (eixo X, Y, Z).

“posicoesIniciais[i][j] = pos[j]”, salva essas posições na matriz posicoesIniciais.

```
WbNodeRef caixa[QtddCaixa];
double posicoesIniciais[QtddCaixa][3];
for (i = 0; i < QtddCaixa; i++) {
  char nomeCaixa[10];
  sprintf(nomeCaixa, "CAIXA%02d", i);
  caixa[i] = wb_supervisor_node_get_from_def(nomeCaixa);
  if (caixa[i] == NULL)
    printf("Falha ao carregar a posição da %s\n", nomeCaixa);
  else {
    const double *pos = wb_supervisor_node_get_position(caixa[i]);
    for (j = 0; j < 3; j++)
      posicoesIniciais[i][j] = pos[j];
  }
}
```

Nesse trecho temos:

“if (!caixaMovida)”, verifica se nenhuma caixa foi marcada como movida ainda. Assim que uma for detectada, essa checagem não será feita mais (evita redundância e acelera a simulação).

Um laco for que itera por todas as caixas (exceto a CAIXA00, aparentemente ignorada de propósito).

“wb_supervisor_node_get_position(caixa[i])” Lê a posição atual da caixa i no espaço 3D.

Para cada eixo (X, Y, Z), calcula a diferença entre a posição atual e a posição inicial.

Usa fabs (valor absoluto) para detectar qualquer deslocamento.

Se a diferença for maior que o DELTA (definido como 0.002), considera que a caixa foi movida.

```c
while (wb_robot_step(TIME_STEP) != -1) {
  for (i = 0; i < 256; i++) texto[i] = 0;

  if (!caixaMovida) {
    for (i = 1; i < QtddCaixa; i++) {
      if (caixa[i] != NULL) {
        const double *posAtual = wb_supervisor_node_get_position(caixa[i]);
        for (j = 0; j < 3; j++) {
          if (fabs(posAtual[j] - posicoesIniciais[i][j]) > DELTA) {
            caixaMovida = true;
            printf("CAIXA%02d FOI MOVIDA!\n", i + 1);
            break;
          }
        }
        if (caixaMovida) break;
      }
    }
  }
```