utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

INTELIGÊNCIA ARTIFICIAL
2024/2025

Professor

Paulo Oliveira

Eduardo Pires

RELATÓRIO

ASPIRADORES EM NETLOGO

TRABALHO PRATICO 1 -> AGENTES

79881 - David Fidalgo
78800 - Tiago Carvalho

2024/2025

Indíce

Indíce de Figuras

1. Resumo

Este trabalho visa descrever o desenvolvimento de uma simulação em NetLogo para estudar a interação entre um agente de limpeza e três agentes poluentes.

Numa primeira fase, foi criado um robô que mapeia e limpa o ambiente, enfrentando desafios como a localização de contentores e a gestão da bateria, além de limpar o ambiente de forma eficiente.

Numa segunda fase, foram implementadas melhorias para aproximar a simulação à realidade. O robot apresenta ajustes energéticos e vários comportamentos conforme o tipo de lixo e obstáculos encontrados.

As simulações mostraram a eficiência de ambos os robôs em diferentes cenários, superando desafios técnicos e apresentado a eficácia das estratégias desenvolvidas para a economia da bateria e melhorar a limpeza em ambientes poluídos.

2. Introdução

A automação de tarefas de carácter diário tem evoluído significativamente com o uso de máquinas e robôs, especialmente em áreas que exigem contante manutenção, como a limpeza de espaços. Neste contexto, a simulação de sistemas multiagente surge como uma ferramenta eficaz para estudar e entender a interação entre diferentes agentes que atuam em um mesmo ambiente.

Este trabalho prático tem como objetivo desenvolver um modelo de simulação, utilizando o NetLogo, que reproduza o comportamento de agentes responsáveis pela limpeza e poluição de um espaço.

A proposta fornecida pelos docentes da cadeira de Inteligência Artificial baseia-se em um cenário simplificado onde um agente de limpeza interage com três agentes poluidores. A simulação visa representar o processo de degradação do espaço causado pelos Polluters e a restauração do espaço realizado pelo Cleaner.

A simulação, não só auxilia na análise das iterações entre agentes, mas também possibilita a análise e experimentação de várias estratégias para melhorar a eficiência do processo de limpeza em um ambiente poluído.

3. Enquadramento Teórico

No contexto deste trabalho, foi utilizado a plataforma NetLogo para simular o comportamento entre agentes de limpeza (Cleaners) e agentes poluentes (Polluters) em um determinado ambiente.

Ambiente:

- Superfície composta por células (patches);
- Todas as células estão limpas e têm a mesma cor;
- É um espaço físico onde os agentes movimentam e interagem entre eles.

***Cleaners*:**

- Localizado inicialmente no canto inferior esquerdo do ambiente;
- Perde uma unidade de energia por movimento;
- Recarrega a bateria no posto de carregamento, localizado no ponto de partida;
- Limpa células com resíduos e armazena os detritos;
- Capacidade de transporte de detritos ajustável;
- Descarrega detritos em contentores (*Containers*), cuja quantidade é ajustável e localização são aleatórios.

***Polluters*:**

- Três agentes poluidores que entram no ambiente em pontos aleatórios;
- Movimentam-se de forma aleatória;
- Depositam resíduos em células limpas, mudando a cor dos *patches* conforme o tipo de resíduo;
- A decisão de depositar resíduos é determinada por uma função probabilísticas, que são ajustáveis para cada *Polluter*.

Interação entre *Cleaner* e *Polluters*:

- *Cleaner* limpa as células poluídas pelos *Polluters*;
- Os *Polluters* continuam a depositar resíduos enquanto se movimentam;
- O *Cleaner* deve gerir a sua energia e a capacidade de armazenamento enquanto limpa o ambiente;
- A eficiência do *Cleaner* depende da sua capacidade de localizar contentores e recarregar a bateria sem interrupções.

4. Robot1

a. Interface

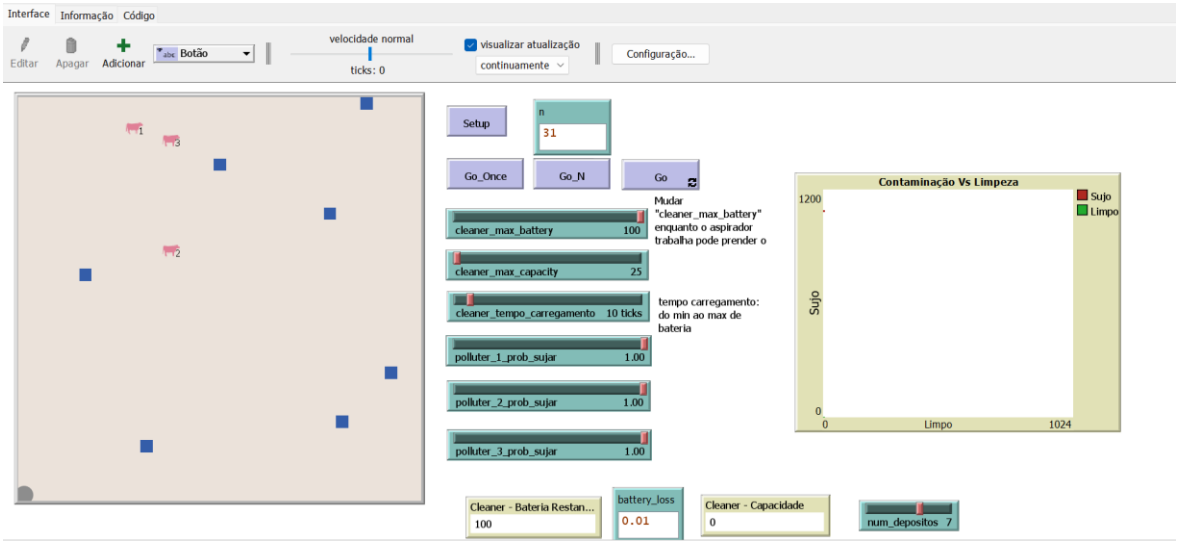

Figura 1-> Robot: Interface NETLOGO

- **Setup:** Reset ao mundo;
- **Go_Once:** Os agentes efetuam um movimento;
- **Go_N:** Os agentes efetuam N movimentos, sendo N adaptável na entrada;
- **Go:** Os agentes realizam de uma forma continua os seus movimentos;
- **Cleaner_max_battery:** Bateria máxima que o cleaner contém;
- **Cleaner_max_capacity:** Capacidade máxima que o cleaner contém;
- **Cleaner_tempo_carregamento:** Ticks necessários para carregar o cleaner;
- **Polluter_1_prob_sujar, Polluter_2_prob_sujar, Polluter_3_prob_sujar:** A probabilidade de cada polluter de poluir;
- **Cleaner – Bateria Restante:** Bateria atual do cleaner;
- **Cleaner – Capacidade:** Capacidade atual do cleaner;
- **Num_depositos:** Numero de depósitos existente no mundo;
- **Contaminação Vs Limpeza:** Grafico que mostra o evoluir a contaminação/limpeza do mundo. Exemplo:

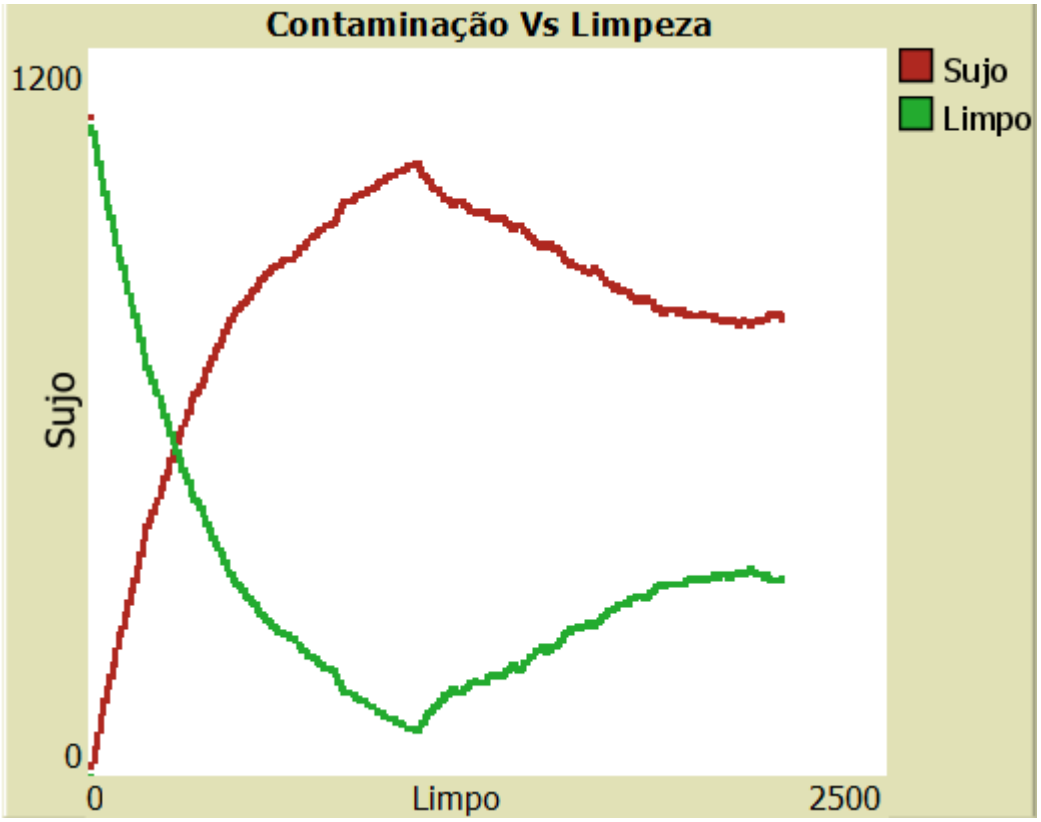

Figura 2-> Robot 1: Gráfico Contaminação Vs Limpeza

b. Adversidades ultrapassadas durante o projeto

1. Localização do Robot (Cleaner)

```
;; MOVER (EM FUNCAO DOS HEADINGS)
if cleaner_stop = 0[
  if patch-ahead 1 != nobody [
    if [pcolor] of patch-at-heading-and-distance heading 1 = cor_objetos [
      right 180
      right random 90 - random 90
    ]
    if battery > 0 [fd 1]
  ]
  ifelse battery <= 0 [
    set battery 0
  ][
    set battery battery - cleaner_consumption_battery
  ]
]
```

Figura 3-> Robot 1: Algoritmo de movimentação cleaner

Para detetar a sua localização o robô recorre ao comando ***patch-here***, que devolve ao nosso agente *cleaner* (robô aspirador) o *patch* onde este se encontra.

Visto que esta abordagem obedece à regra de que o *cleaner* só pode ter perceção de apenas um *patch* à sua volta (**comando *neighbors***), pareceu-nos a mais adequada.

O nosso agente *cleaner* usa os seguintes comandos chave para determinar onde se encontra e quem são os seus vizinhos:

Comandos chave:

- patch-here
- patch-ahead
- patch-at-heading-and-distance
- neighbors

2. Localização dos Contentores:
   - Numa fase inicial, o cleaner vai ter o modo mapeamento ativado, que serve para localizar todos os contentores existentes, que são todos os *patches-here* que se encontrem com o *pcolor* = *blue*;

```
to Modo_Map
  if xcor = min-pxcor[ ;chega borda esquerda
    ifelse patch-ahead 1.5 = nobody[ ; se é parede
      set heading 0
    ][
      set heading 90
    ]
  ]
  if xcor = max-pxcor[ ;chega borda direita
    ifelse patch-ahead 1.5 = nobody[ ; se é parede
      set heading 0
    ][
      set heading -90
    ]
  ]
  if round xcor = max-pxcor and round ycor = max-pycor [
    set cleaner_map FALSE
  ]
  if [pcolor] of patch-here = blue [
    let coordenadas_depositos (list round xcor round ycor)
    set depositos lput coordenadas_depositos depositos
  ]
end
```

Figura 4-> Robot 1: Função Modo_Map

- Quando chega ao limite superior, ele começa o processo de limpeza;
- Quando o cleaner precisar de descarregar os detritos, ele calcula, na lista **depósitos**, procura as coordenadas mais próximas à localização atual do cleaner.

```
let menor-distancia (max-pxcor * 3)
let target-patch one-of depositos
let j 0
foreach depositos [
  let coordenada item j depositos
  let x item 0 coordenada
  let y item 1 coordenada
  let distancia-atual distancexy x y
  if distancia-atual < menor-distancia [
    set menor-distancia distancia-atual
    set target-patch patch x y
  ]
  set j j + 1
]
if target-patch != nobody [
  ask cleaner cleaner_atual [
    face target-patch  ; Faz o cleaner olhar para o patch-alvo
  ]
]
```

Figura 5-> Robot 1: Algoritmo de procura o deposito mais próximo

- Caso o utilizador queira mudar o número de contentores, o cleaner continua a mapear caso encontre um *patch* de cor *blue.* Caso o cleaner se desloque a um sítio que deveria conter um deposito e esse *patch* não tiver com a cor devida, essa coordenada é removida da lista de depósitos.

```
ask patch-here[
  if pcolor != blue [
    let coordenadas_depositos (list round pxcor round pycor)
    if member? coordenadas_depositos depositos[
      set depositos remove coordenadas_depositos depositos
    ]
  ]
]
```

Figura 6-> Robot 1: Remover deposito já não existente

3. Posto de Carregamento

- O posto de carregamento encontra se no canto inferior esquerdo;
- O Cleaner só se desloca para o posto de carregamento só quando faltar 50 movimentos;

```
ask cleaners[
  ifelse battery <= 50 * battery_loss[;; dirigir ao posto de carregamento quando so faltarem 50 movimentos
    if last_cleaning_location = [0 0][;; aspirador guarda sitio onde estava a aspirar até ter de ir carrega
      set last_cleaning_location (list round xcor round ycor)
      if ticks > tick_bug_fix [set last_cleaning_location [-15 -15] set tick_bug_fix tick_bug_fix + 10000];
    ]
    facexy item 0 posto_carregamento item 1 posto_carregamento ;; código direcçao à bateria
```

Figura 7-> Robot 1: Verificação e deslocação para posto de carregamento

- O Clenear, após o número de ticks de carregamento, volta para o local onde estava a limpar.

c. Simulação 1

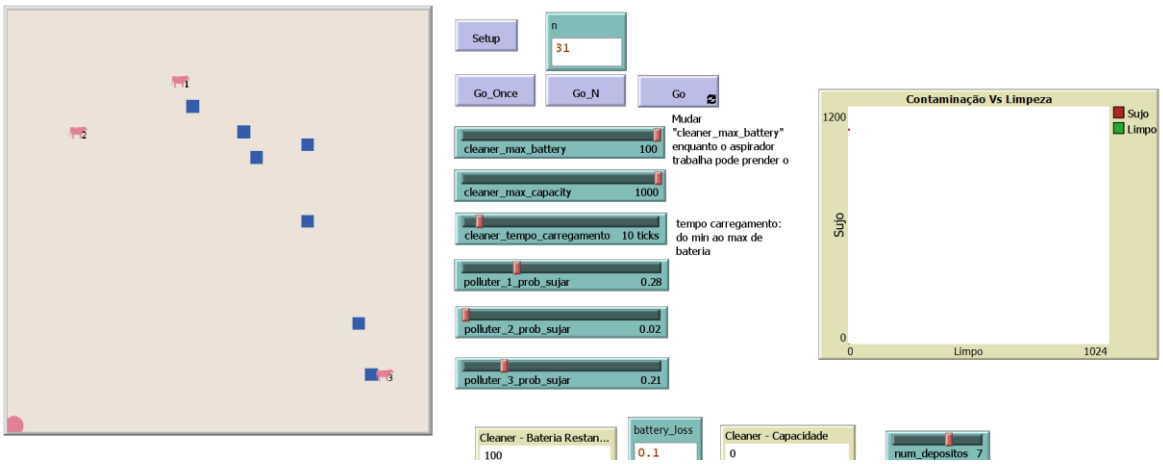

Figura 8-> Robot1: Simulação - Setup

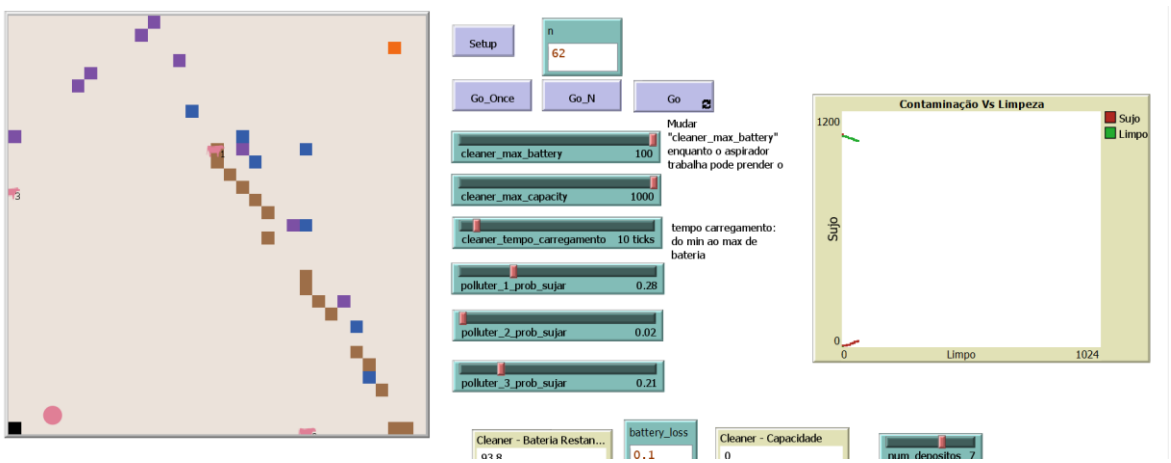

Figura 9-> Robot 1: Simulação - Go-N

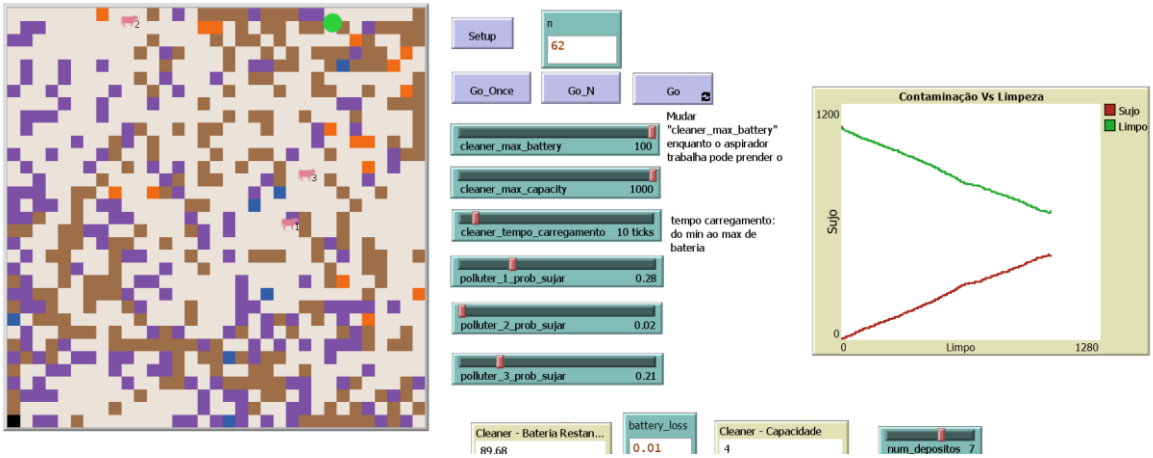

Figura 10-> Robot 1: Simulação - Mapping Efetuado

- Após mapear os contentores, começa a limpar o ambiente.

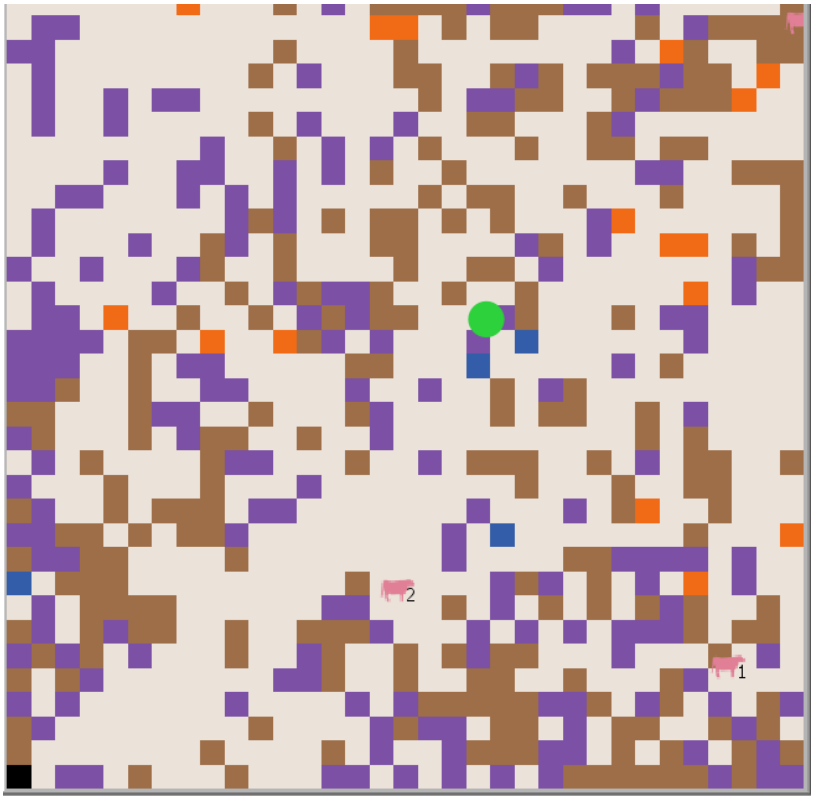

Figura 11-> Robot1: Simulação - Desaparecimento de Contentor

- Caso o utilizador tenha alterado o número de contentores, o *robot* só reconhece as alterações quando passar no *patch* onde surgiu/desapareceu um depósito.
- Caso o *Robot* vá lá e encontre o espaço vazio (ou seja, a cor seja diferente de *blue*), ele vai apagar da sua memória as coordenadas onde estaria o contentor.
- Quando o *robot* vá a um espaço onde esteja um contentor e ainda não estiver na memória, o robot guarda na memória as coordenadas do *patch* onde se encontra.

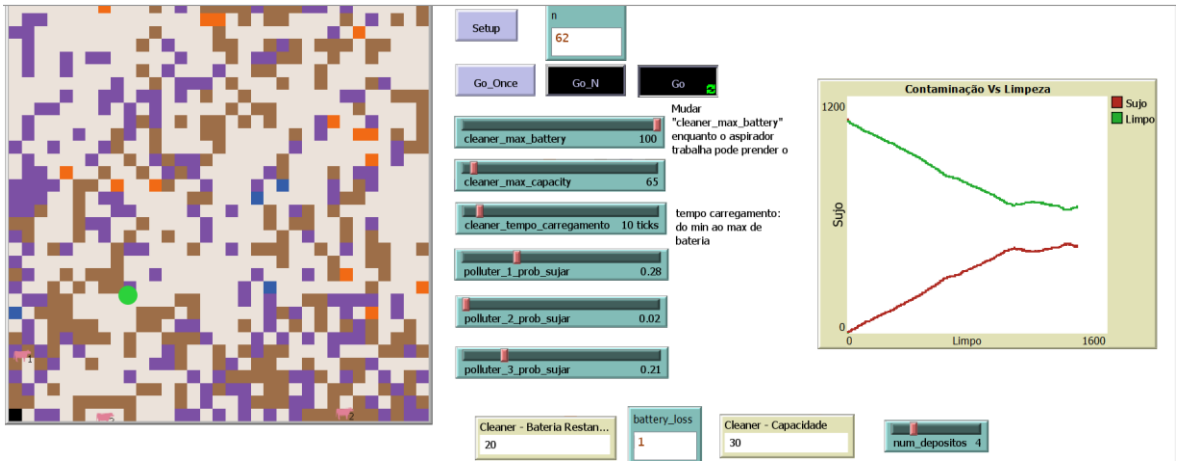

Figura 12-> Robot 1: Simulação - Direção Posto Carregamento

- Quando faltar 50 movimentos (*battery_loss* * 50 movimentos), o *Robot* direciona para o posto de carregamento, evitando assim uma possível paragem devido à falta de bateria.
- O Robot guarda o local onde ficou antes de deslocar-se ao posto de carregamento.

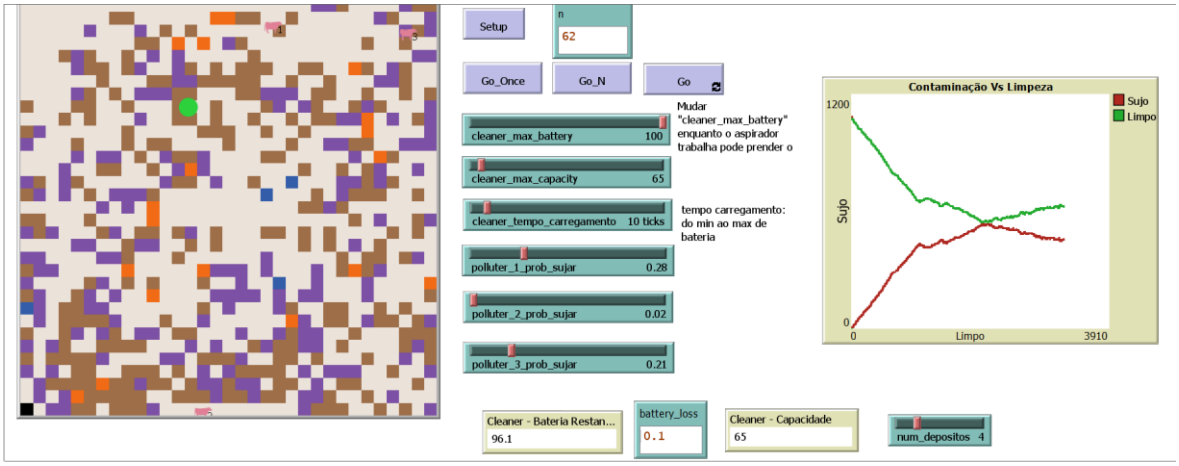

Figura 13-> Robot 1: Simulação – Recuperação do local

- Após o carregamento, o Robot desloca-se para o local onde se encontrava antes de deslocar para o posto de carregamento.

5. Robot 2

a. Interface

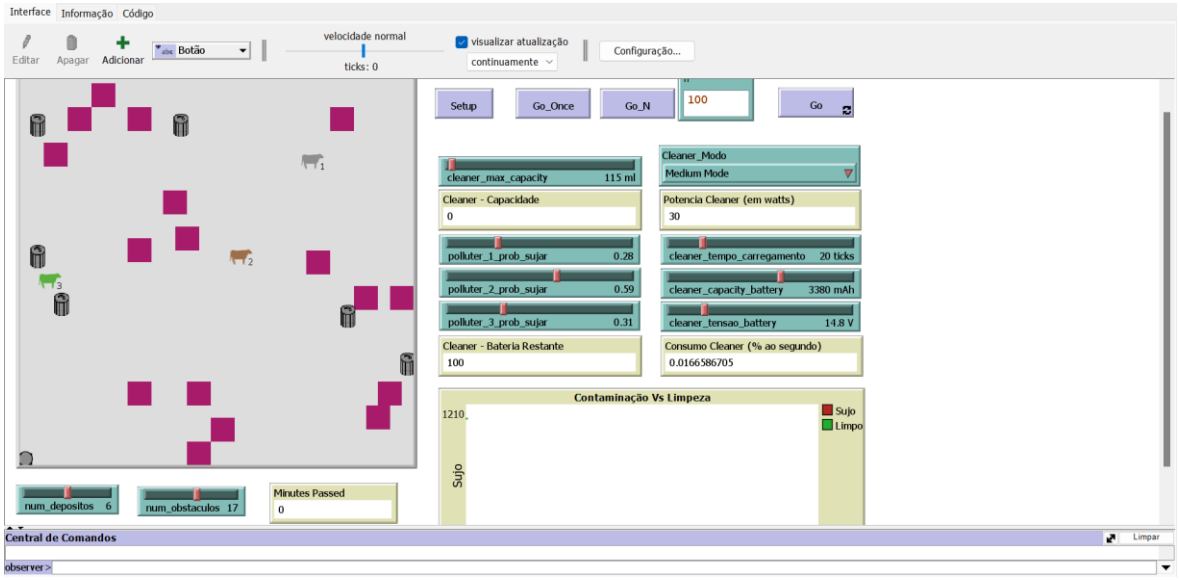

Figura 14-> Robot 2: Interface

Nesta fase, mantivemos a base da fase transata, aplicando melhorias, aproximando o ambiente dos agentes o mais próximo da realidade.

- **Cleaner_Modo:** O cleaner vai ter 3 modos de funcionamento: Eco Mode, Medium Mode e Full Mode;
- **Potencia Cleaner (em W):** Valor, em Watts, da Potencia do Cleaner;
- **Cleaner_capacity_battery:** Capacidade da bateria, em miliampere-hora, do Cleaner;
- **Cleaner_tensao_battery:** Tensão, em Volts, da bateria do Cleaner;
- **Consumo Cleaner:** Consumo, por tick, a cada movimento do Cleaner;
- **Cleaner_max_capacity (Alteração):** Capacidade máxima, em ml, do Cleaner;
- **Num_obstaculos:** Numero de obstáculos do ambiente.

b. Implementações para aproximação com a Realidade

1. Inserção de modelo de bateria ao Cleaner

```
to Config_Battery
  set eco "Eco Mode"
  set med "Medium Mode"
  set turbo "Full Mode"

  ask cleaners[
    if Cleaner_Modo = eco [
      ;; comportamento para o modo Eco
      set cleaner_potencia_battery 10
    ]
    if Cleaner_Modo = med [
      ;; comportamento para o modo Médio
      set cleaner_potencia_battery 30
    ]
    if Cleaner_Modo = turbo [
      ;; comportamento para o modo Full
      set cleaner_potencia_battery 50
    ]
    let cleaner_corrente_battery (cleaner_potencia_battery / cleaner_tensao_battery) * 1000;; isto é a corrente em mA
    let cleaner_ma_segundo_battery cleaner_corrente_battery / 3600 ;; aqui o gasto de mA por segundo
    set cleaner_consumption_battery (cleaner_ma_segundo_battery / cleaner_capacity_battery) * 100
  ]

end
```

Figura 15-> Robot 2: Função Config_Battery

- Cada um dos modos vai ter diferentes valores de potência;
- Conforme o Modo do Cleaner, da capacidade da bateria e da tensão, é calculado o consumo da bateria por segundo (ou tick).

2. Diferentes tons de poluição

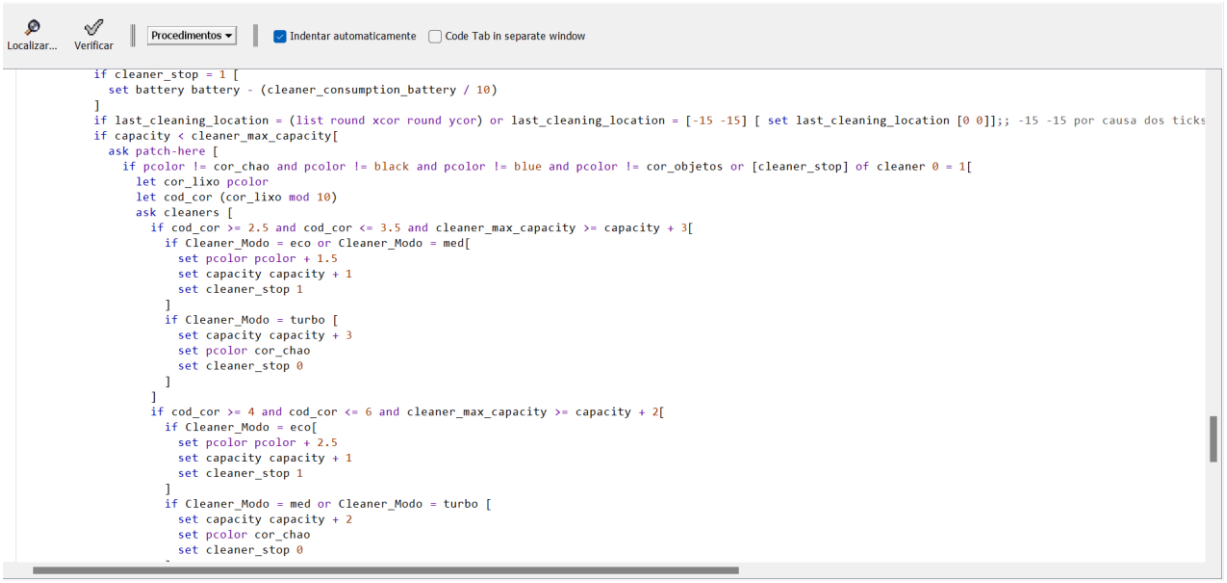

Figura 16-> Robot 2: Comportamento Cleaner perante poluição

- Os *polluters* vão inserir ‘lixos’ de diferentes tons de uma determinada cor;
- O *Cleaner* vai ter diferentes comportamentos, dependendo do modo em que se encontrar:
    - **Full Mode:**
        - Neste modo, o *cleaner* vai apenas demorar 1 *tick* a limpar o *patch* que se encontra, com todo o tipo de lixo.
    - **Medium Mode:**
        - Os lixos considerados “leves” e “normais”, o *cleaner* só necessita de 1 *tick* para limpar o *patch* em questão;
        - Os lixos considerados “pesados”, o cleaner vai aumentar o valor do pcolor até o cleaner encontrar se em condições de limpar facilmente o lixo.
    - **Eco Mode:**
        - Os lixos considerados “leves”, o cleaner necessita de 1 tick para limpar o patch em questão;
        - Os livros considerados “normais” e “pesados”, o *cleaner* vai aumentar o valor do *pcolor* até o lixo que se encontrar no *patch* for considerado “leve”.
- Cada lixo tem diferente peso.

c. Simulação

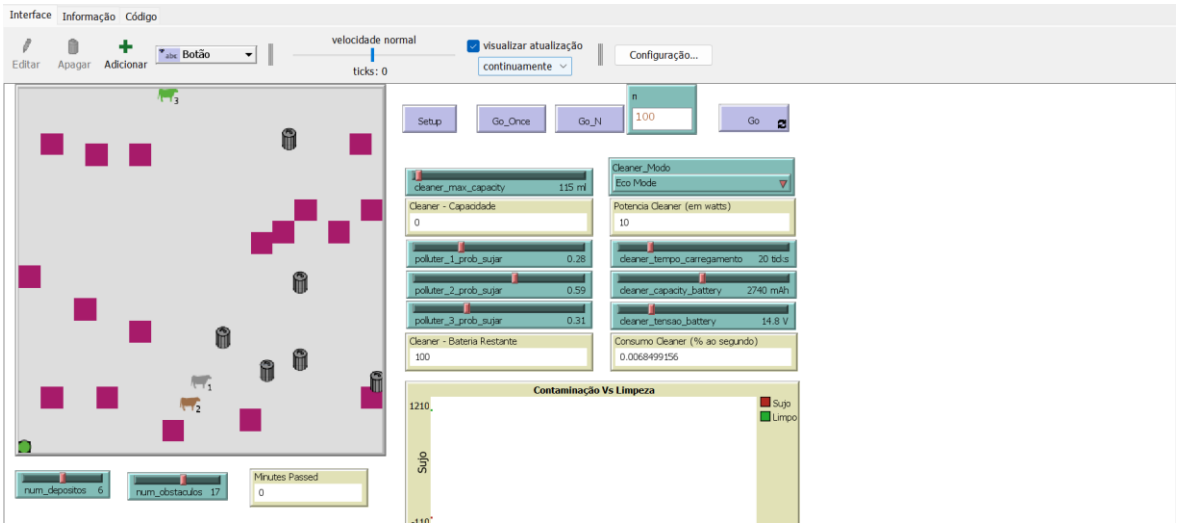

Figura 17-> Robot 2: Simulação – Setup

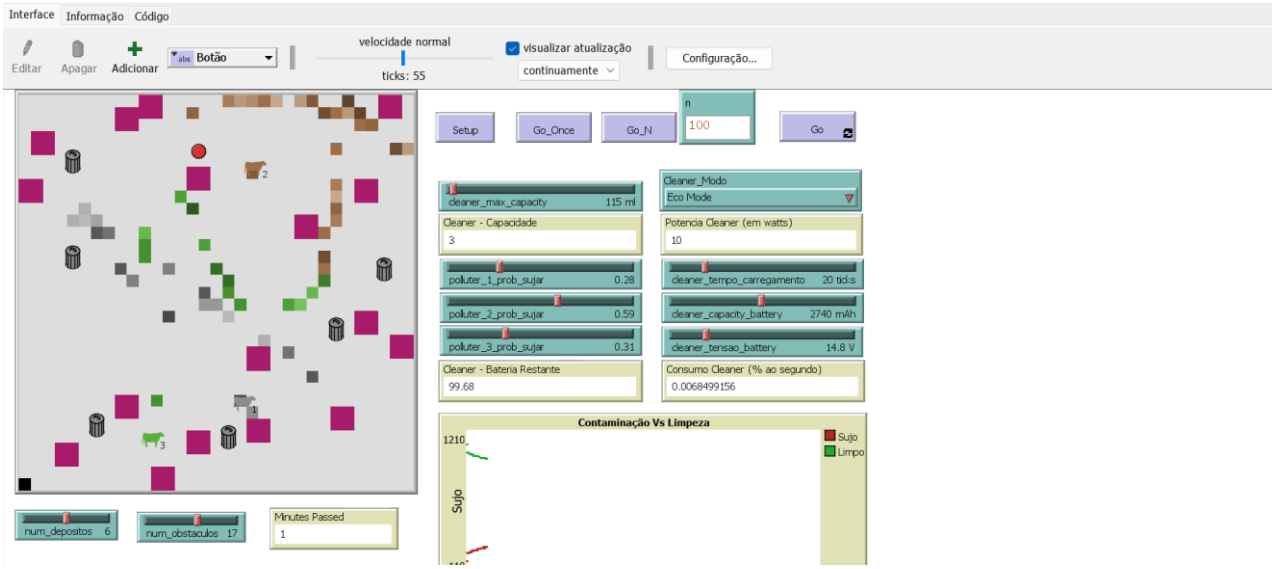

Figura 18-> Robot 2: Simulação – Obstáculos

- Caso esteja em modo limpeza, se o *robot* encontre à sua frente um obstáculo, vai escolher uma direção aleatória e continua a limpeza.
- Caso o *robot* esteja a dirigir-se para uma determinada localização *(last_cleaning_location, posto_carregamento, depósitos)* ele procura contornar o obstáculo escolhendo uma direção (esquerda ou direita).

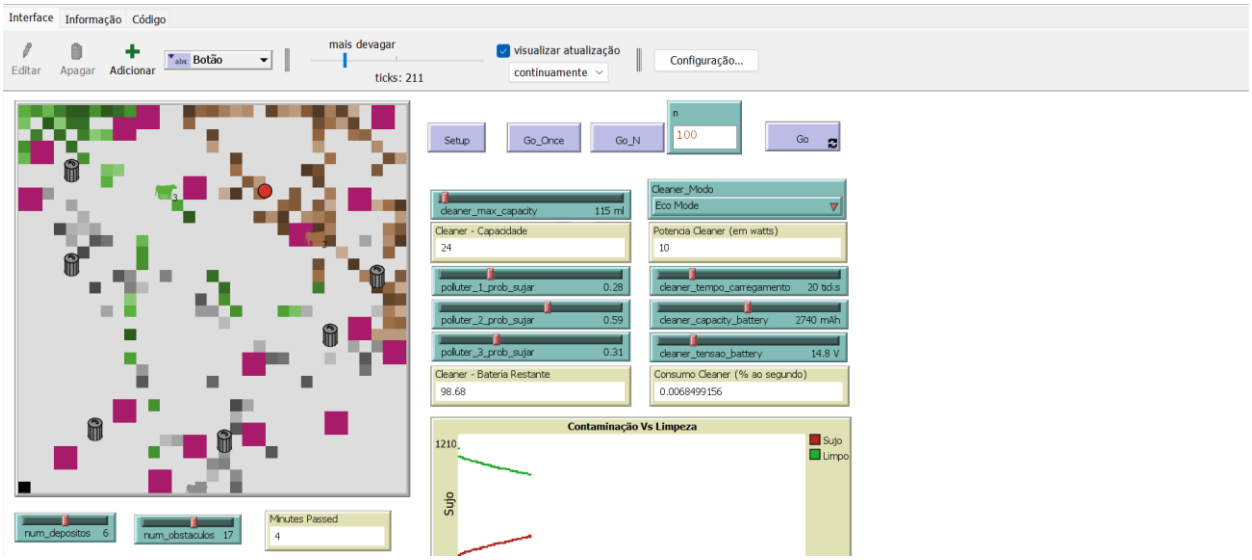

Figura 19-> Robot 2: Simulação - Limpeza Patch

- Quando o *robot* encontra um *patch* de uma intensidade de cor alta e ao mesmo tempo encontrar se nos modos mais económicos, ele demora alguns *ticks* a mais e a cada um tira intensidade à cor.

6. Resultados

Na fase 1, o *Robot* não apresentou problemas, executando todas as tarefas conforme especificado no protocolo. O *Robot* conseguiu mapear e limpar o ambiente de forma eficiente, localizando os contentores e gerir a bateria sem dificuldades. A várias simulações feitas demonstraram que o robô cumpriu todos os requisitos estabelecidos, validando a eficácia dos algoritmos implementados para a navegação e limpeza do espaço.

Já na fase 2, foi possível implementar diferentes modelos de bateria, ajustando a sua capacidade, tensão a potência do *Robot.* O consumo de energia do robô aproxima-se da realidade, refletindo um comportamento realista em termos de eficiência e durabilidade da bateria. Em 5 simulações, o Robot opera por uma média de 80 horas, sem necessitar de intervenção humana para recarregar, sendo essas ocasiões este tido ficado preso em obstáculos e consequentemente tendo perdido a carga.

7. Conclusão

Este trabalho prático conseguiu modelar com sucesso a interação entre agentes responsáveis pela poluição (Polluters) e pela limpeza (Cleaner) de um espaço, utilizando NetLogo. Através de várias simulações, foi possível, não só reproduzir o comportamento esperado dos agentes, mas também ultrapassar desafios técnicos relacionados à localização e eficiência dos mesmos.

A integração de modos de funcionamento do Cleaner, o mapeamento de depósitos e o desenvolvimento de estratégias para economizar bateria mostram uma aproximação da simulação à realidade. O Cleaner tem um desempenho ajustável conforme diferentes modos de operação, variando o consumo de energia e a eficiência com base no tipo de lixo e obstáculos encontrados.

Em suma, estamos satisfeitos pelo trabalho elaborado, mesmo com a ideia de que haveria sempre espaço para melhorias.

8. Bibliografia

- https://youtu.be/O7ozptNs1FY?si=MSywmYDwbmLPsnCb (consultado a 7/10/2024)
- https://stackoverflow.com/questions/36019543/turtles-move-to-nearest-patch-of-a-certain-color-how-can-this-process-be-sped (consultado a 12/10/2024)
- OpenAI. (2024). *ChatGPT: AI Language Model*. https://www.openai.com
- WILENSKY, U. *NetLogo User Manual* – NetLogo Dictionary. Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2023. Disponível em: https://ccl.northwestern.edu/netlogo/docs/dictionary.html.

Anexo Robot 1

```
;; Definição de variáveis globais e raças
globals [cor_chao posto_carregamento depositos tick_bug_fix]
breed [cleaners cleaner]
breed [polluters polluter]
cleaners-own [battery capacity recharge_time last_cleaning_location cleaner_map]
polluters-own [prob_sujar polluter_corlixo]

;; Modo mapear depósitos (zig-zag até percorrer todo o espaço da sala)
;; NÃO ASPIRA
to Modo_Map
  if xcor = min-pxcor [
    ifelse patch-ahead 1.5 = nobody [ ; Se é parede
      set heading 0
    ][
      set heading 90
    ]
  ]
  if xcor = max-pxcor [
    ifelse patch-ahead 1.5 = nobody [ ; Se é parede
      set heading 0
    ][
      set heading -90
    ]
  ]
  if round xcor = max-pxcor and round ycor = max-pycor [
    set cleaner_map FALSE
```

```
    ]
    if [pcolor] of patch-here = blue [
      let coordenadas_depositos (list round xcor round ycor)
      set depositos lput coordenadas_depositos depositos
    ]
  end

  ;; Setup: Limpa o ambiente, cria e introduz os agentes, e faz o reset do tempo.
  to setup
    clear-all
    reset-ticks
    set tick_bug_fix 10000 ;; Para evitar loops infinitos

    set cor_chao 39
    ask patches [ set pcolor cor_chao ]

    set posto_carregamento [-16 -16]
    ask patch item 0 posto_carregamento item 1 posto_carregamento [ set pcolor black
]

    ;; Criação de depósitos
    let i 1
    ask patches [
      set i count patches with [pcolor = blue]
      if pcolor = cor_chao and i < num_depositos and one-of [pcolor] of neighbors4 !=
blue [
        set pcolor blue
      ]
```

```
]

;; Criação de agentes
create-cleaners 1
create-polluters 3

;; Configuração dos cleaners
ask cleaners [
  set depositos []
  set shape "circle"
  set size 1.5
  setxy item 0 posto_carregamento item 1 posto_carregamento
  set battery cleaner_max_battery
  set capacity 0
  set last_cleaning_location [0 0]
  set cleaner_map TRUE
  set heading 90
]

;; Configuração dos polluters
ask polluters [
  set shape "cow"
  set size 1.5
  set color pink
  set label-color black
  set label who
  setxy random-pxcor random-pycor
```

```
  ]
end

;; go_once: Permite que os agentes circulem no mundo de forma aleatória (um só tick).
to go_once
  ;; Atualiza probabilidades dos sliders
  ask polluter 1 [
    set prob_sujar polluter_1_prob_sujar
    set polluter_corlixo 33
  ]
  ask polluter 2 [
    set prob_sujar polluter_2_prob_sujar
    set polluter_corlixo 53
  ]
  ask polluter 3 [
    set prob_sujar polluter_3_prob_sujar
    set polluter_corlixo 3
  ]

  ;; Atualiza depósitos
  let i 1
  ask patches [
    set i count patches with [pcolor = blue]
    if pcolor = cor_chao and i < num_depositos [
      set pcolor blue
    ]
    if i > num_depositos and pcolor = blue [
```

```
      set pcolor cor_chao
    ]
  ]

  ;; Ações dos cleaners
  ask cleaners [
    let cleaner_atual who
    ask patch-here [
      if pcolor != blue [
        let coordenadas_depositos (list round pxcor round pycor)
        if member? coordenadas_depositos depositos [
          set depositos remove coordenadas_depositos depositos
        ]
      ]
    ]

    ;; Modo carregar bateria
    if battery > cleaner_max_battery [ set battery cleaner_max_battery ]
    ask patch-here [
      ifelse pcolor = black and ([battery] of cleaner cleaner_atual < cleaner_max_battery) [
        ask cleaners [
          let battery_a_cargar battery + (cleaner_max_battery / cleaner_tempo_carregamento)
          ifelse battery_a_cargar > cleaner_max_battery [
            set battery cleaner_max_battery
          ][
            set battery battery + (cleaner_max_battery / cleaner_tempo_carregamento)
```

```
    ]
  ]
][
  ;; Verificar a bateria e definir movimentação
  ask cleaners [
    ifelse battery <= 50 * battery_loss [
      if last_cleaning_location = [0 0] [
        set last_cleaning_location (list round xcor round ycor)
        if ticks > tick_bug_fix [
          set last_cleaning_location [-15 -15]
          set tick_bug_fix tick_bug_fix + 10000
        ]
      ]
      facexy item 0 posto_carregamento item 1 posto_carregamento
    ][
      ifelse capacity >= cleaner_max_capacity [
        ;; Modo ir depositar resíduos
        ifelse [pcolor] of patch-here = blue [
          let coordenadas_depositos (list round xcor round ycor)
          if not member? coordenadas_depositos depositos [
            set depositos lput coordenadas_depositos depositos
          ]
          set capacity 0
        ][
          let menor-distancia (max-pxcor * 3)
          let target-patch one-of depositos
          let j 0
```

```
        foreach depositos [
          let coordenada item j depositos
          let x item 0 coordenada
          let y item 1 coordenada
          let distancia-atual distancexy x y
          if distancia-atual < menor-distancia [
            set menor-distancia distancia-atual
            set target-patch patch x y
          ]
          set j j + 1
        ]
        if target-patch != nobody [
          ask cleaner cleaner_atual [ face target-patch ]
        ]
      ]
    ][
      ;; Movimento para limpar
      ifelse last_cleaning_location != [0 0] [
        facexy item 0 last_cleaning_location item 1 last_cleaning_location
      ][
        if patch-ahead 1 = nobody and cleaner_map = FALSE [set heading random
360]
      ]
    ]
  ]
  if cleaner_map = FALSE [
    fd 1
    set battery battery - battery_loss
```

```
                if last_cleaning_location = (list round xcor round ycor) or last_cleaning_location = [-15 -15] [
                  set last_cleaning_location [0 0]
                ]
                if capacity < cleaner_max_capacity [
                  ask patch-here [
                    if pcolor != cor_chao and pcolor != black and pcolor != blue [
                      set pcolor cor_chao
                      ask cleaners [ set capacity capacity + 1 ]
                    ]
                  ]
                ]
              ]
              if cleaner_map = TRUE [
                Modo_Map
                fd 1
                if last_cleaning_location = (list round xcor round ycor) or last_cleaning_location = [-15 -15] [
                  set last_cleaning_location [0 0]
                ]
                set battery battery - 0.01
              ]
            ]
          ]
        ]

        ;; Ações dos polluters
```

```
ask polluter 1 [
  if patch-ahead 1 = nobody [ set heading random 360 ]
  fd 1
  if (random 100 < prob_sujar * 100) [
    ask patch-here [
      if pcolor = cor_chao [ set pcolor [polluter_corlixo] of polluter 1 ]
    ]
  ]
]

ask polluter 2 [
  if patch-ahead 1 = nobody [ set heading random 360 ]
  fd 1
  if (random 100 < prob_sujar * 100) [
    ask patch-here [
      if pcolor = cor_chao [ set pcolor [polluter_corlixo] of polluter 2 ]
    ]
  ]
]

ask polluter 3 [
  if patch-ahead 1 = nobody [ set heading random 360 ]
  fd 1
  if (random 100 < prob_sujar * 100) [
    ask patch-here [
      if pcolor = cor_chao [ set pcolor [polluter_corlixo] of polluter 3 ]
    ]
```

```
    ]
  ]
  tick
end

;; go_n: Executa o processo 'go_once' repetidamente n vezes
to go_n
  repeat n [ go_once ]
end
```

Anexo Robot 2

```
;; globals

globals[cor_chao cor_objetos posto_carregamento depositos lista_objetos tick_bug_fix tipo_lixo num_polluters cleaner_max_battery eco med turbo]

breed[cleaners cleaner]

breed[polluters polluter]

breed[containers container]

breed[obstacles obstacle]

cleaners-own[battery capacity recharge_time last_cleaning_location cleaner_consumption_battery cleaner_potencia_battery cleaner_stop]

polluters-own[prob_sujar]


to Config_Battery
  set eco "Eco Mode"
  set med "Medium Mode"
  set turbo "Full Mode"


  ask cleaners [
    if Cleaner_Modo = eco [
      ;; comportamento para o modo Eco
      set cleaner_potencia_battery 10
    ]
    if Cleaner_Modo = med [
      ;; comportamento para o modo Médio
      set cleaner_potencia_battery 30
    ]
    if Cleaner_Modo = turbo [
      ;; comportamento para o modo Full
```

```
    set cleaner_potencia_battery 50
  ]
  let cleaner_corrente_battery (cleaner_potencia_battery / cleaner_tensao_battery) * 1000 ;; isto é a corrente em mA
  let cleaner_ma_segundo_battery cleaner_corrente_battery / 3600 ;; aqui o gasto de mA por segundo
  set cleaner_consumption_battery (cleaner_ma_segundo_battery / cleaner_capacity_battery) * 100
 ]
end

; setup, cujo programa permita: limpar o ambiente; criar e introduzir no mundo os agentes e fazer o reset do tempo.
to setup
 clear-all
 reset-ticks
 set tick_bug_fix 10000 ; de 10000 em 10000 ticks reset da last_cleaning location senao ele pode ficar preso num loop de ir de ponta a ponta
 set tipo_lixo (list (range 2.5 7.5 0.5)(range 32.5 37.5 0.5)(range 52.5 57.5 0.5))
 set cor_chao 8.5
 set cor_objetos magenta
 set cleaner_max_battery 100
 ask patches [
  set pcolor cor_chao
 ]
 set posto_carregamento [-16 -16]
 ask patch item 0 posto_carregamento item 1 posto_carregamento [;; caso mude tamanho do world
  set pcolor black
```

```
]

;; padrões do dicionário do netlogo
create-cleaners 1
create-polluters 3

ask cleaners [
  set shape "vaccum"
  set size 2.5
  let canto_inferior_esquerdo (list min-pxcor min-pycor) ; origem do cleaner (posto de carregamento)
  setxy (item 0 canto_inferior_esquerdo ) (item 1 canto_inferior_esquerdo) ;; criado no canto inferior esquerdo
  set battery 100
  set capacity 0
  set last_cleaning_location [0 0]
  set cleaner_stop 0
]
Config_Battery

ask polluters [
  set shape "cow"
  set size 2
  set color white
  set label-color black
  set label who
  setxy random-pxcor random-pycor
]
```

```
ask polluter 1 [ set color 5 ]
ask polluter 2 [ set color 35 ]
ask polluter 3 [ set color 55 ]

;; criacao de depositos
let i 1
set depositos []
ask patches [
  set i count patches with [pcolor = blue]
  if pcolor = cor_chao and i < num_depositos [;; evitar depositos juntos (fica confuso)
    if all? neighbors4 [pcolor = cor_chao] [
      set pcolor blue
      sprout-containers 1
      set depositos fput (list pxcor pycor) depositos
    ]
  ]
]
ask containers [
  set shape "garbage can"
  set size 2
  set color grey
]

;; criacao obstaculos
set i 0
set lista_objetos []
ask patches [
```

```
        set i count patches with [pcolor = cor_objetos]
        if pcolor = cor_chao and i < num_obstaculos * 4 and pxcor < 16 and pycor > -16 [
          let object1 (list (list pxcor pycor) (list (pxcor + 1) pycor) (list pxcor (pycor - 1)) (list (pxcor + 1) (pycor - 1)))
          let pode_gerar true
          foreach object1 [coord ->
            let x item 0 coord
            let y item 1 coord
            if [pcolor] of patch x y != cor_chao [
              set pode_gerar false
            ]
          ]
          if pode_gerar [
            foreach object1 [coord ->
              let x first coord
              let y last coord
              ask patch x y [
                set pcolor cor_objetos ;; Replace cor_objetos with the desired color
              ]
            ]
            set lista_objetos fput object1 lista_objetos
          ]
        ]
      ]
    end
```

; go_once, cujo programa permita: que os agentes circulem no mundo de forma aleatória (um só tick);

```
to go_once
  ;; atualizar probabilidades dos sliders
  ask polluters [
    if color = 5  [set prob_sujar polluter_1_prob_sujar]
    if color = 35 [set prob_sujar polluter_2_prob_sujar]
    if color = 55 [set prob_sujar polluter_3_prob_sujar]
  ]
  Config_Battery

  ;; atualizar depositos
  let i 1
  ask patches [
    set i count patches with [pcolor = blue]
    if pcolor = cor_chao and i < num_depositos [
      set pcolor blue
      set depositos lput (list pxcor pycor) depositos
      sprout-containers 1 [
        set shape "garbage can"
        set size 2
        set color grey
      ]
    ]
    if i > num_depositos and pcolor = blue [
      set pcolor cor_chao
      ask containers-here [die]
      set depositos remove (list pxcor pycor) depositos
    ]
```

```
]

;; ações do cleaner
ask cleaners [
  let cleaner_atual who ;; para permitir mais cleaners e usar o codigo abaixo

  ;; modo carregar
  if battery > cleaner_max_battery [ set battery cleaner_max_battery ]
  ask patch-here [
   ifelse pcolor = black and ([battery] of cleaner cleaner_atual < cleaner_max_battery) [
     ask cleaners [
      let battery_a_cargar battery + (cleaner_max_battery / cleaner_tempo_carregamento) ;; battery_a_cargar é o cálculo de quanto a bateria vai carregar em um tick
      ifelse battery_a_cargar > cleaner_max_battery [
       set battery cleaner_max_battery
      ][
       set battery battery + (cleaner_max_battery / cleaner_tempo_carregamento) ;; por cada tick para 100 max é tipo: 100/10 = 10% a cada tick
      ]
     ]
    ][
     ;; HEADINGS E AS SUAS CONDICOES
     ;; 1º verificar a bateria (modelo Robot1 dirige-se ao posto quando chega a uma certa percentagem)
     ask cleaners [
      ifelse battery <= 100 * cleaner_consumption_battery [ ;; dirigir ao posto de carregamento quando so faltarem 50 movimentos
```

```
if last_cleaning_location = [0 0] [ ;; aspirador guarda sitio onde estava a aspirar até ter de ir carregar bateria
  set last_cleaning_location (list round xcor round ycor)
  if ticks > tick_bug_fix [set last_cleaning_location [-15 -15] set tick_bug_fix tick_bug_fix + 10000] ;; senao ele fica la em cima e nao volta.... porque nao tem movimentos random suficiente para voltar para baixo
 ]
 facexy item 0 posto_carregamento item 1 posto_carregamento ;; código direcçao à bateria
][
 ifelse capacity >= cleaner_max_capacity [
  ;; modo ir depositar
  ifelse [pcolor] of patch-here = blue [
   set capacity 0 ;; esvazia capacidade toda (ia melhorar mas melhor guardar para fase 2)
  ][
   let target-patch min-one-of (patches in-radius 40 with [pcolor = blue]) [distance myself] ;; (apenas esta linha é) solução stackoverflow: https://stackoverflow.com/questions/36019543/turtles-move-to-nearest-patch-of-a-certain-color-how-can-this-process-be-sped
   if target-patch != nobody [
    ask cleaner cleaner_atual [
     face target-patch ;; direcionar para o deposito
    ]
   ]
  ]
 ][
  ;; movimento modo aspirar
  ifelse last_cleaning_location != [0 0] [ ;; voltar ao local anterior
```

```
                facexy item 0 last_cleaning_location item 1 last_cleaning_location ;; TO UPGRADE: virar caso bata enquanto vai para sitio dele
              ][
                ;; aspirar área desconhecida
                if patch-ahead 1 = nobody or ([pcolor] of patch-ahead 1 != cor_chao and [pcolor] of patch-ahead 1 != black) [
                  right random 360
                ]
              ]
              ;; mudar cor (limpar o chão)
              ifelse pcolor = cor_chao [
                if capacity + 1 <= cleaner_max_capacity [
                  set pcolor cor_lixo_cleaned
                  set capacity capacity + 1
                ]
              ][
                set last_cleaning_location [0 0] ;; caso tenha batido num obstáculo ou já tenha limpo sítio anterior
              ]
            ]
          ]
        ]
      ]
    ]
    ;; baixa de bateria (dps do ciclo em q limparam o lixo)
    ask cleaners [
      set battery battery - cleaner_consumption_battery ;; cleaner_consumption_battery é a variável q depende da potência
```

```
    if battery < 0 [ set battery 0 ]
   ]
  ]

  ;; ações dos polluters
  ask polluters [
   if random 100 <= prob_sujar [
    ask patch-here [
     ifelse pcolor = cor_chao [
      let lixo random-float 100 ;; random para decidir tamanho do lixo
      ifelse lixo <= 40 [
       set pcolor item 0 tipo_lixo
      ][
       ifelse lixo <= 70 [
        set pcolor item 1 tipo_lixo
       ][
        set pcolor item 2 tipo_lixo
       ]
      ]
     ][
      ifelse pcolor = item 0 tipo_lixo or pcolor = item 1 tipo_lixo or pcolor = item 2 tipo_lixo [
       ;; NÃO fazer nada pois já tem lixo
      ][
       right random 360 ;; não largar lixo em cima de objetos ou do depósito
      ]
     ]
    ]
```

```
    ][
      right random 30
      forward 1
    ]
  ]
  tick
end
```

; go_n, cujo programa permita: que os agentes circulem no mundo de forma aleatória (vários ticks);

```
to go_n
  repeat n [
    go_once
  ]
end
```