

PDS - Processo de Desenvolvimento de Software

The Bug is on the Table

Bruno Rodrigues
Iago Rodrigues
Jonathan Rufino
Matheus Andrade
Yago Regis

03-07-2015

Lista de Figuras

| | | |
|---|---|----|
| 1 | Base para acoplamento dos veículos. | 20 |
| 2 | Garra para derrubar o galho e capturar objetos. | 21 |
| 3 | Braço para acionamento do mecanismo das ondas. | 22 |

Lista de Tabelas

| | | |
|---|--|---|
| 1 | Relação de pontos por missão. | 4 |
| 2 | Relação de dificuldade das missões | 5 |

Sumário

| | | |
|----------|---|-----------|
| 1 | Missões | 4 |
| 1.1 | Ordem de Pontuação das Missões | 4 |
| 1.2 | Ordem de Dificuldade das Missões | 4 |
| 1.3 | Definindo o Problema | 4 |
| 2 | Caminhão de Suplementos, Ambulância, Sinal de Evacuação e Isolamento de Construção | 6 |
| 2.1 | Objetivo da missão | 6 |
| 2.2 | Passos envolvidos | 6 |
| 2.3 | Pseudo código | 7 |
| 2.4 | Código NXC | 8 |
| 3 | Galho da Árvore, Animais e Equipamentos | 10 |
| 3.1 | Objetivo da Missão | 10 |
| 3.2 | Passos envolvidos | 10 |
| 3.3 | Pseudo-código | 10 |
| 3.4 | Código NXC | 10 |
| 4 | Tsunami | 12 |
| 4.1 | Objetivo da missão | 12 |
| 4.2 | Passos envolvidos | 12 |
| 4.3 | Pseudo-código | 12 |
| 4.4 | Código NXC | 12 |
| 5 | Família, Água, Segurança, Animais, Suplementos e Equipamentos, Zona de Segurança | 14 |
| 5.1 | Objetivo da missão | 14 |
| 5.2 | Passos envolvidos | 14 |
| 5.3 | Pseudo-código | 14 |
| 5.4 | Código NXC | 14 |
| 6 | Musica - Game of Thrones Main Theme | 16 |
| 6.1 | Objetivo | 16 |
| 6.2 | Código NXC | 16 |
| 7 | API | 18 |
| 7.1 | Objetivo da missão | 18 |
| 7.2 | Passos envolvidos | 18 |
| 7.3 | Pseudo-código | 18 |
| 7.4 | Código NXC | 18 |
| 8 | Garras | 20 |

1 Missões

O documento do desafio Fúria da Natureza, nos da diretrizes à respeito de todas as missões que podem ser realizadas. Cada uma possui sua descrição, condições finais e a respectiva pontuação.

1.1 Ordem de Pontuação das Missões

As missões foram listadas em ordem decrescente de pontuação, de forma que a pontuação considerada seja a maior possível de cada uma.

| Pontuação | Missão |
|-----------|-----------------------------|
| 66 | Família |
| 31 | Obstáculos |
| 30 | Galho da Árvore |
| 30 | Pista de Pouso |
| 30 | Sinal de Evacuação |
| 30 | Teste de Isolamento de Base |
| 30 | Avião de Carga |
| 25 | Ambulância |
| 25 | Casa Elevada |
| 25 | Construção de Código |
| 25 | Zona de Segurança |
| 20 | Camiñhão de Suplementos |
| 20 | Relocação de Construção |
| 20 | Tsunami |
| 18 | Segurança |
| 15 | Água |
| 15 | Animais |
| 4 | Suplementos e Equipamentos |

Tabela 1: Relação de pontos por missão.

1.2 Ordem de Dificuldade das Missões

Aqui temos uma lista das missões ordenadas por dificuldade, da mais fácil para a mais difícil, desta forma pode-se realizar um cruzamento de informações entre pontuação e dificuldade afim de que a equipe faça uma escolha das missões que realizará, priorizando de acordo com o esforço necessário e a pontuação recebida.

1.3 Definindo o Problema

Algumas missões são completamente independentes uma das outras, entretanto, existem missões que devem seguir uma sequência lógica para que não

| Dificuldade | Missão |
|-------------|-----------------------------|
| FÁCIL | Zona de Segurança |
| FÁCIL | Caminhão de Suplementos |
| FÁCIL | Tsunami |
| MÉDIO | Família |
| MÉDIO | Galho da Árvore |
| MÉDIO | Pista de Pouso |
| MÉDIO | Avião de Carga |
| MÉDIO | Água |
| MÉDIO | Animais |
| MÉDIO | Suplementos e Equipamentos |
| DIFÍCIL | Obstáculos |
| DIFÍCIL | Teste de Isolamento da Base |
| DIFÍCIL | Casa Elevada |
| DIFÍCIL | Construção de Código |
| DIFÍCIL | Relocação de Construção |
| DIFÍCIL | Segurança |

Tabela 2: Relação de dificuldade das missões

atrapalhem as demais. Cada missão deve possuir um planejamento antes de ser desenvolvida e executada.

2 Caminhão de Suplementos, Ambulância, Sinal de Evacuação e Isolamento de Construção

2.1 Objetivo da missão

- O caminhão de suplementos está tocando o tapete na região amarela;
- A ambulância está na área amarela;
- Todas as rodas da ambulância estão tocando o tapete;
- O sinal está obviamente em pé (não precisa ser na vertical), mantido no lugar apenas pelo atrito da viga com o tapete;
- Nenhum parte do modelo de missão está sendo tocado pelo robô ou qualquer obstáculo estratégico;
- A ambulância está na área amarela; Todas as rodas da ambulância estão tocando o tapete.
- O prédio oeste está intacto: seus 4 segmentos estão a 90° do tapete, e “perfeitamente” alinhados.
- O edifício leste está obviamente danificado.
- *Nada está tocando nenhum dos prédios exceto a base de rolamento.
- *Nada nunca tocou nenhum dos prédios exceto a base de rolamento.
- O dano foi causado unicamente pelo movimento da base de rolamento.
- (*Exceção: Segmentos caídos do edifício leste podem tocar o tapete e/ou o edifício oeste por acaso.)

2.2 Passos envolvidos

Sair da base
Pegar o caminhão
Pegar a ambulância
Sseguir até a área amarela
Empurrar o Sinal de Evacuação
Destruir o prédio direto das torres
Voltar para a base

2.3 Pseudo código

1. Mova em frente 38cm
2. Pare de mover
3. Vire à direita 90 graus
4. Mova em frente 20cm (para chegar ao caminhão)
5. Pegue o caminhão
6. Mova em frente 30 cm (para chegar à ambulância)
7. Vire à esquerda 15 graus
8. Pegue a ambulância
9. Mova em frente 90 cm (para chegar na área azul)
10. Pare de mover
11. Mova para trás 35cm
12. Pare de mover
13. Vire à direita 45 graus
14. Mova em frente 40cm (para empurrar a placa de sinalização)
15. Pare de mover
16. Mova para trás 20cm
17. Pare de mover
18. Vire 175 graus
19. Mova em frente 90cm
20. Pare de mover
21. Vire à esquerda 30 graus
22. Mova em frente 40cm (para chegar à base)
23. Encerre o programa

2.4 Código NXC

```
1  /*
2  * Initial Position:
3  * Right side of the "claw", aligned at the end of the "M" by inside.
4  */
5 #include "theBugAPI.h"
6
7 task main() {
8
9     // Moving to Supply Truck and Ambulance.
10    move(38, POWER_NORMAL, FORWARD);
11    turn(85, POWER_NORMAL, RIGHT);
12    Wait(1);
13    move(50, POWER_NORMAL, FORWARD);
14    turn(9, POWER_LOW, LEFT);
15    Wait(1);
16    move(20, POWER_NORMAL, FORWARD);
17
18    // Moving to Yellow Mark.
19    move(100, POWER_NORMAL, FORWARD);
20    turn(80, POWER_HIGH, LEFT);
21    Wait(1);
22    move(25, POWER_NORMAL, FORWARD);
23
24    // Moving to Evacuation Signal.
25    move(20, POWER_NORMAL, BACKWARD);
26    turn(30, POWER_NORMAL, LEFT);
27    Wait(1);
28    move(40, POWER_NORMAL, BACKWARD);
29    turn(35, POWER_NORMAL, RIGHT);
30    Wait(1);
31    move(21, POWER_NORMAL, FORWARD);
32
33    // Moving to Isolamento de Construção.
34    move(30, POWER_NORMAL, BACKWARD);
35    turn(90, POWER_NORMAL, LEFT);
36    Wait(1);
37    move(30, POWER_NORMAL, FORWARD);
38    move(14, POWER_HIGH, FORWARD);
39
40    //Moving to base
41    move(15, POWER_NORMAL, BACKWARD);
42    turn(83, POWER_NORMAL, LEFT);
43    Wait(1);
44    move(150, POWER_HIGH, FORWARD);
```


3 Galho da Árvore, Animais e Equipamentos

3.1 Objetivo da Missão

- O galho leste da árvore está mais próximo do tapete do que os cabos elétricos estão;
- A árvore e o modelo de missão dos cabos elétricos estão para cima, retos, tocando o tapete;
- Ao menos um animal está com pelo menos uma pessoa em uma região colorida;
- Ao menos um elemento que não é água está numa região colorida vermelha ou amarela (12 elementos possíveis: walkie talkie, bateria, gerador, 2 combustíveis, grão, pão, remédio, rádio, lanterna, motocicleta e capacete).

3.2 Passos envolvidos

Sair da Base

Ir ao lado do galho da árvore

Derrubar o galho da árvore

Pegar os animais e equipamentos da região

Retornar a Base

3.3 Pseudo-código

Mova em frente 14 cm

Vire à esquerda 87 graus

Mova em frente 67 cm

Pare de mover

Levante a garra 70 graus

Mova para trás 15 cm

Vire para esquerda 90 graus

Mova para frente 18 cm

Vire para direita 90 graus

Mova para frente 19 cm

Desça a garra 70 graus

Mova para trás 86 cm

Encerre o programa

3.4 Código NXC

```
1  /*
2  *  * Initial Position:
```

```

3  * Right side of the "claw", aligned at the end of the "M" by left-inside.
4  */
5 #include "theBugAPI.h"
6
7 task main() {
8
9    // Moving to the tree.
10   move(70, POWER_NORMAL, FORWARD);
11   claw_control(10, POWER_HIGH, BACKWARD);
12   claw_control(90, POWER_HIGH, BACKWARD);
13   claw_control(100, POWER_HIGH, FORWARD);
14
15   // Moving to catch the animals.
16   move(30, POWER_NORMAL, BACKWARD);
17   turn(85, POWER_NORMAL, RIGHT);
18   Wait(1);
19   move(22, POWER_NORMAL, BACKWARD);
20   turn(90, POWER_NORMAL, LEFT);
21   Wait(1);
22   move(39, POWER_NORMAL, FORWARD);
23   claw_control(70, POWER_NORMAL, BACKWARD);
24
25   // Moving to base.
26   move(70, POWER_HIGH, BACKWARD);
27 }
```

4 Tsunami

4.1 Objetivo da missão

- O galho leste da árvore está mais próximo do tapete do que os cabos elétricos estão.
- A árvore e o modelo de missão dos cabos elétricos estão para cima, retos, tocando o tapete.

4.2 Passos envolvidos

Sair da base
Ir até a estrutura do tsunami
Acionar o mecanismo
Retornar a base

4.3 Pseudo-código

Mova em frente até que o sensor leia uma distância de 20cm ou menos
Levante a garra Xcm
Retorne a base

4.4 Código NXC

```
1  /*
2   * Initial Position:
3   * Aligned straight with the tsunami.
4   */
5  #include "theBugAPI.h"
6
7  task main() {
8
9      // Indicates the port to which the sensor is connected.
10     SetSensorUltrasonic(IN_2);
11
12     // The robot will move up to a certain distance.
13     while(true) {
14         // Delay to read sensor.
15         Wait(1000);
16
17         // print the value read by the sensor.
18         ClearScreen();
19         NumOut(0, 0, SensorUS(IN_2));
20 }
```

```
21         if(SensorUS(IN_2) < 23) {
22             Off(OUT_BC);
23             break;
24         }
25         else {
26             OnFwd(OUT_BC, POWER_LOW);
27         }
28     }
29
30     Wait(50);
31     claw_control(60, POWER_NORMAL, BACKWARD);
32
33     // Moving to base.
34     move(70, POWER_HIGH, BACKWARD);
35 }
```

5 Família, Água, Segurança, Animais, Suplementos e Equipamentos, Zona de Segurança

5.1 Objetivo da missão

- Ao menos duas pessoas estão juntas em uma área colorida.
- Ao menos uma pessoa está com uma água (engarrafada) na mesma região.
- Ao menos uma pessoa está na região colorida vermelha ou amarela.
- Ao menos um animal está com pelo menos uma pessoa em uma região colorida.
- Ao menos um elemento que não é água está numa região colorida vermelha ou amarela.
- O robô está na região vermelha no final da partida.

5.2 Passos envolvidos

Sair da base

Pegar a pessoa ao lado da casa

Ir para a região vermelha

5.3 Pseudo-código

Saia da base

Ande Xcm

Vire a direita Xgraus

Ande Xcm

Vire a esquerda Xgraus

Ande Xcm

Levante a garra Xgraus

Ande Xcm para tras

Vire Xgraus a direita

Ande Xcm para frente

Vire Xgraus para a direta

Ande em frente até o sensor detectar a linha vermelha

5.4 Código NXC

```
1  /*
2   * Initial Position:
3   * Right side of the "claw", aligned at the end of the "M" by left-inside.
```

```

4  /*
5  #include "theBugAPI.h"
6
7 #define COLORSENSOR_SENSOR_2
8 #define RED 5
9
10 task main() {
11
12     move(43, POWER_HIGH, FORWARD);
13     turn(85, POWER_HIGH, RIGHT);
14     Wait(1);
15     move(67, POWER_HIGH, FORWARD);
16     turn(78, POWER_LOW, LEFT);
17     Wait(1);
18     move(22, POWER_HIGH, FORWARD);
19     claw_control(55, POWER_NORMAL, BACKWARD);
20     move(15, POWER_HIGH, BACKWARD);
21     turn(85, POWER_HIGH, RIGHT);
22     Wait(1);
23     move(80, POWER_HIGH, FORWARD);
24     turn(44, POWER_HIGH, RIGHT);
25     Wait(1);
26
27 // Turn on the RGB sensor
28     SetSensorColorFull(S2);
29
30     while(Sensor(S2) != RED) {
31         OnFwd(OUT_BC, POWER_NORMAL);
32     }
33
34     PlaySound(GOT_RAINS_OF_CASTAMERE);
35 }

```

6 Musica - Game of Thrones Main Theme

6.1 Objetivo

O professor propôs que os alunos que conseguissem fazer o robô tocar alguma música em suas missões iriam ganhar 5 pontos. Assim, o grupo escolheu a música tema da série Game of Thrones para o robô tocar.

Foi implementado apenas a primeira parte, visto que a música em seguida fica mais complexa e o grupo conseguia fazer o robô executar apenas uma nota por vez, e que, por tanto, iria perder a qualidade da música.

Foram criadas constantes baseada nas figuras musicais encontradas na partitura e o tempo indicado na partitura para facilitar a implementação. Foi criado constante para a primeira oitava que o robô consegue tocar, para que pudesse servir de apoio para implementação de funções que encontravam as frequências das notas sem precisar escrever elas diretamente na função “PlayToneEx”, contudo, essa última parte não foi implementada.

6.2 Código NXC

```
1  #define VOL 3
2  #define A3 220
3  #define AS3 233
4  #define B3 247
5  #define C 262
6  #define CS 277
7  #define D 294
8  #define DS 311
9  #define E 330
10 #define F 349
11 #define FS 370
12 #define G 392
13 #define GS 415
14 #define A 440
15 #define AS 466
16 #define B 494
17 #define SEMIBREVE_PONTUADA 4000
18 #define SEMINIMA_PONTUADA 1000
19 #define SEMINIMA 666
20 #define COLCHEIA 333
21 #define SEMICOLCHEIA 166
22
23 sub background_strings() {
24     PlayToneEx(659, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
25     PlayToneEx(440, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
```

```

26     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
27     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
28     PlayToneEx(659, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
29     PlayToneEx(440, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
30     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
31     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
32     PlayToneEx(659, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
33     PlayToneEx(440, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
34     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
35     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
36     PlayToneEx(659, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
37     PlayToneEx(440, COLCHEIA, VOL, FALSE); Wait(COLCHEIA);
38     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
39     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
40 }
41
42 task main() {
43     // Main theme of game of thrones.
44     PlayToneEx(392, SEMINIMA_PONTUADA, VOL, FALSE); Wait(SEMINIMA_PONTUADA);
45     PlayToneEx(264, SEMINIMA_PONTUADA, VOL, FALSE); Wait(SEMINIMA_PONTUADA);
46     PlayToneEx(311, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
47     PlayToneEx(349, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
48     PlayToneEx(392, SEMINIMA, VOL, FALSE); Wait(SEMINIMA);
49     PlayToneEx(262, SEMINIMA, VOL, FALSE); Wait(SEMINIMA);
50     PlayToneEx(311, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
51     PlayToneEx(349, SEMICOLCHEIA, VOL, FALSE); Wait(SEMICOLCHEIA);
52     PlayToneEx(294, SEMIBREVE_PONTUADA, VOL, FALSE); // Wait(SEMIBREVE_PONTUADA);
53
54     background_strings();
55 }
```

7 API

- 7.1 Objetivo da missão
- 7.2 Passos envolvidos
- 7.3 Pseudo-código
- 7.4 Código NXc

```
1  #define PI 3.1416
2
3  #define WHEEL_RADIUS 2.16
4  #define WHEEL_DIAMETER 4.32
5  #define CAR_RADIUS 6.65
6
7
8  #define RIGHT -1
9  #define LEFT 1
10
11 #define FORWARD 1
12 #define BACKWARD -1
13
14 #define POWER_NORMAL 75
15 #define POWER_LOW 50
16 #define POWER VERY_LOW 25
17 #define POWER_HIGH 100
18
19 #define COMPLETE_ROTATION 360
20 #define HALF_ROTATION 180
21
22
23
24 sub turn(int angle, int power, int direction){
25     int car_arc = (angle * PI * CAR_RADIUS) / HALF_ROTATION; // Measured in centimeter
26     int amount_turns = (HALF_ROTATION * car_arc) / (PI * WHEEL_RADIUS); // Measured in centimeter
27
28     RotateMotorExPID(OUT_BC, power, amount_turns, 100 * direction,
29                       true, true, 40, 40, 90);
30 }
31 sub move(int distance, int power, int direction){
32     int angle = (distance * COMPLETE_ROTATION * direction) / (PI * WHEEL_DIAMETER);
33     RotateMotorEx(OUT_BC, power, angle, 0, true, true);
34 }
35
36 sub claw_control(int angle, int power, int direction){
37     RotateMotor(OUT_A, power, angle * direction);
```

```
38 }
```

```
1 #define WHEEL_RADIUS 2.16
2 #define WHEEL_DIAMETER 4.32
3 #define CAR_RADIUS 6.65
4
5
6 #define RIGHT -1
7 #define LEFT 1
8
9 #define FORWARD 1
10 #define BACKWARD -1
11
12 #define POWER_NORMAL 75
13 #define POWER_LOW 50
14 #define POWER_VERY_LOW 25
15 #define POWER_HIGH 100
16
17 #define COMPLETE_ROTATION 360
18 #define HALF_ROTATION 180
19
20
21
22 sub turn(int angle, int power, int direction){
23     int car_arc = ( angle * PI * CAR_RADIUS )/ HALF_ROTATION; // Measured in centimeter
24     int amount_turns = (HALF_ROTATION * car_arc) /(PI * WHEEL_RADIUS); // Measured in centimeter
25
26     RotateMotorExPID(OUT_BC, power, amount_turns, 100 * direction,
27                         true, true, 40, 40, 90);
28 }
29 sub move(int distance, int power, int direction){
30     int angle = (distance * COMPLETE_ROTATION * direction)/(PI * WHEEL_DIAMETER);
31     RotateMotorEx(OUT_BC, power, angle, 0, true, true);
32 }
33
34 sub claw_control(int angle, int power, int direction){
35     RotateMotor(OUT_A, power, angle * direction);
36 }
```

8 Garras

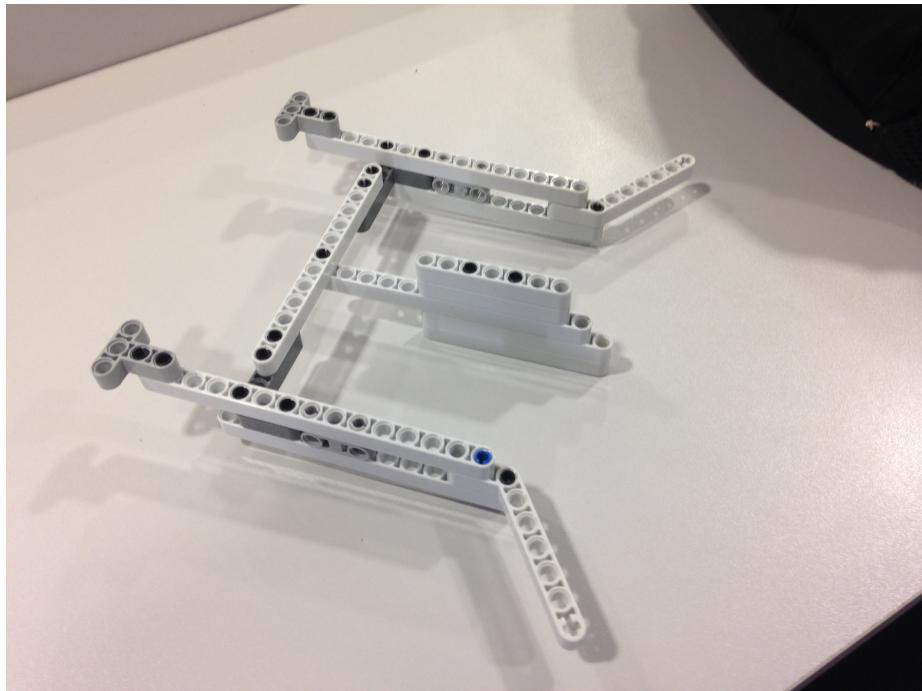


Figura 1: Base para acoplamento dos veículos.



Figura 2: Garra para derrubar o galho e capturar objetos.



Figura 3: Braço para acionamento do mecanismo das ondas.