



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

XFCE - Relatório

Autores: Dylan Guedes, Geovanni Oliveira, Hebert Douglas,
Tallys Martins, Victor Carvalho, Vitor Meireles

Brasília, DF
2015



Dylan Guedes, Geovanni Oliveira, Hebert Douglas, Tallys Martins, Victor
Carvalho, Vitor Meireles

XFCE - Relatório

Relatórios finais sobre os experimentos da
disciplina Fundamentos de Redes e Compu-
tadores.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Brasília, DF

2015

Sumário

1	INTRODUÇÃO	3
2	OBJETIVOS	5
2.1	Objetivo Geral	5
2.2	Objetivos específicos	5
3	EQUIPE	7
4	ROBÔ	9
4.1	Estrutura do Robô	9
4.2	Linguagem adotada	9
5	MISSÕES	11
5.1	Descrição	11
5.2	Contagem dos pontos	11
5.3	Priorização	11
5.4	Justificativa	12
6	SOLUÇÃO EM SOFTWARE DAS MISSÕES	13
	Referências	17

1 Introdução

A disciplina de Princípios de Robótica Educacional tem como orientação a competição já consolidada Lego Nature's Fury, que contém regras, tarefas, objetivos e critérios para nortear a competição. Portanto, a disciplina se baseia numa competição com os mesmos moldes do Nature's Fury, existindo assim vários times, que são completamente livres para definir quais tarefas irão fazer, de que maneira o robô será escrito, quais critérios irá atender e quais não, bem como outras coisas descritas no guia do Nature's Fury.

Contudo, todos os resultados obtidos necessitam ser documentados, e este é o objetivo deste documento. Aqui serão descritas as atividades/missões cumpridas pelo time, justificativas (a respeito do porquê da prioridade de determinadas tarefas), explicações de como se chegar ao resultado obtido, bem como análises.

2 Objetivos

2.1 Objetivo Geral

O objetivo geral do desafio é construir um robô que execute o maior número de missões (conseguindo assim um maior número de pontos) em um tempo de, no máximo, dois minutos e meio.

2.2 Objetivos específicos

- Construir um robô utilizando peças do kit Lego Mindstorms;
- Executar as missões, afim de se obter êxito;
- Elaborar um relatório descrevendo os passos e explicando as escolhas definidas para cada uma das missões.

3 Equipe

A equipe é formada por 6 alunos da disciplina de Princípios de Robótica Educacional, que estão listados abaixo:

- Dylan Guedes;
- Geovanni Oliveira;
- Hebert Douglas;
- Tallys Martins;
- Victor Carvalho;
- Vitor Meireles.

4 Robô

O robô é montado a partir de peças disponíveis do kit Lego Mindstorms. O kit contém diversas peças de encaixe, três sensores (diferentes entre si) e um cérebro. A montagem do robô é livre, sendo assim, os alunos podem modelar o robô da maneira que lhes for conveniente.

4.1 Estrutura do Robô

A estrutura do robô seguiu o padrão tanque, utilizando para a movimentação as esteiras. O grupo optou por essa modelagem pois, dessa maneira, o robô adquire centro de massa conveniente para as missões, sendo mais flexível a mudanças (do tipo aumentar a velocidade com que ele se movimenta), entre outras coisas. A garra utilizada para determinadas missões é em formato de meio-quadrado, e é fixa (sendo assim, deverá ser trocada para missões futuras). A garra escolhida é grande e foi crucial para as missões concluídas.

4.2 Linguagem adotada

Inicialmente o grupo tinha maior interesse na linguagem Java, principalmente pela parte de orientação a objetos. Contudo, uma escolha posterior foi feita, e a linguagem utilizada atualmente pelo time é a linguagem NXC. A razão é o fato da sintaxe ser familiar à todos os integrantes do grupo (muito similar à linguagem C), e oferece padrões que serão explorados mais pra frente, como por exemplo, a parte de programação paralela usando tasks (similar à utilização de threads).

5 Missões

5.1 Descrição

Nome da missão	Descrição	Pontuação	Dificuldade
Ambulância	Uma ambulância se encontra no mapa do desafio. O objetivo é que o robô empurre a ambulância sem virar ou tombá-la até a parte indicada no mapa, em azul.	25	Média
Caminhão	Um caminhão se encontra no mapa do desafio. O objetivo é que o robô empurre o caminhão sem virar ou tombar até a parte indicada no mapa, em azul.	20	Média
Seta	Uma seta é localizada no mapa. O robô tem como objetivo empurrar a seta, levantando assim a placa.	30	Média
Tsunami	Uma rampa deve sofrer colisão, derrubando assim os canudos armazenados.	20	Fácil
Safe Place	O robô deve chegar com segurança à área vermelha do mapa.	25	Fácil
Uso de sensor	O robô deve usar, de maneira útil, algum dos sensores no auxílio das missões.	25	Média
Música	O robô deve emitir sons em algum momento durante a execução de alguma missão.	5	Fácil
Supplies and Equipment	Os itens presentes no caminhão devem chegar de maneira segura até a área demarcada no mapa.	3*3	Difícil
Safety	Pessoas devem permanecer na área vermelha do mapa durante a execução das missões.	18	Média
Runway	Nada deve tocar a área de evacuação do tapete (área próxima à seta).	30	Média

5.2 Contagem dos pontos

$$T = 40 + 60 + 75 + 5 + 9 + 18 = 207 \quad (5.1)$$

5.3 Priorização

As missões são feitas em uma determinada ordem determinada pelo grupo baseado. Essas escolhas são feitas levando-se em consideração quão perto uma tarefa se encontra de

outra (seguindo assim um fluxo comum, economizando tempo), quão difícil é uma tarefa comparado à quantidade de pontos que ela proporciona, e se era possível realizar tal tarefa (nem todas as tarefas dispõem dos obstáculos neste momento).

A priorização escolhida foi:

1. Tsunami;
2. Sensor auxiliando no Tsunami;
3. Caminhão;
4. Ambulância;
5. Supplies and Equipment;
6. Runway;
7. Safety;
8. Seta;
9. Safe Place;
10. Música.

5.4 Justificativa

Como já citado, seguindo-se um fluxo comum de tarefas, o tempo é otimizado. Começa-se com a tarefa Tsunami (com auxílio do sensor de presença), pois fica próxima à área inicial, permitindo assim que o robô seja resetado, corrigindo assim sua posição e execução do código. O robô então captura a ambulância e em seguida o trator (são próximos um do outro), completa este objetivo chegando à faixa azul, levando junto os suprimentos do caminhão, e em seguida completa-se a missão da seta. Se a missão da seta fosse feita primeiro, o robô teria que voltar até o início do mapa e empurrar de novo, gastando tempo desnecessário (além de estar mais propenso a variações). Por fim, o robô chega com segurança à área vermelha, e emite uma música (*Mario Theme*).

6 Solução em Software das missões

Para a codificação das missões sentiu-se a necessidade de realizar a modularização das implementações o código abaixo foi denominado como *moviment.nxc*, que engloba todas as implementações referentes aos movimentos do robô.

```

1  #define COMPRIMENTO 10.83
2  #define POTENCIA 75
3
4  int defineAngulo(int distancia){
5      int total = distancia * 360;
6      int angulo = total/COMPRIMENTO;
7
8      return angulo;
9  }
10
11 void andarFrente(int distancia){
12     int angulo = defineAngulo(distancia);
13     RotateMotor(OUT_AC, POTENCIA, angulo);
14 }
15
16 void andarTras(int distancia){
17     int angulo = defineAngulo(distancia);
18     RotateMotor(OUT_AC, -POTENCIA, angulo);
19 }
20
21 int girar(int angulo){
22     int converte = (38.5*angulo/180);
23
24     return converte;
25 }
26
27 void virarEsquerda(int angulo){
28     int distancia = girar(angulo);
29     int anguloR = defineAngulo(distancia);
30
31     RotateMotorEx(OUT_AC, POTENCIA, anguloR, -100, true, true);
32 }
33
34 void virarDireita(int angulo){
35     int distancia = girar(angulo);
36     int anguloR = defineAngulo(distancia);
37

```

```

38     RotateMotorEx(OUT_AC, POTENCIA, anguloR, 100, true, true);
39 }
40
41 void baixarGarra(int angulo) {
42     RotateMotor(OUT_B, 50, angulo);
43 }
44
45 void levantarGarra(int angulo) {
46     RotateMotor(OUT_B, 50, -angulo);
47 }

```

A vantagem dessa modularização é que o código das missões fica mais enxuto facilitando a leitura e o desenvolvimento pois a implementação segue a lógica em que cada chamada de função é um passo a ser realizado para completá-las. Abaixo segue o código do arquivo *mission1.nxc* e *mission2.nxc* com as missões realizadas (Trator, Ambulância, Seta, Área vermelha, Tsunami, etc).

```

1  #include "moviment.nxc"
2
3  task main() {
4      andarFrente(40);
5      virarDireita(90);
6      andarFrente(9);
7      virarDireita(12);
8      andarFrente(55);
9      andarFrente(60);
10     virarEsquerda(10);
11     andarFrente(75);
12
13     //mission2 Seta
14     andarTras(70);
15     virarDireita(27);
16     andarFrente(55);
17 }

```

```

1  #include "moviment.nxc"
2  #define NEAR 15 //cm
3
4  void sensorUltrassonico() {
5      SetSensorLowSpeed(IN_4);
6      while(true) {
7          OnFwd(OUT_AC, 50);
8          while(SensorUS(IN_4) > NEAR)
9              Off(OUT_AC);

```



```

10         // OnRev(OUT_AC,100);
11         Wait(800);
12     }
13 }
14
15 void sensorUltrassonico1() {
16     SetSensorLowSpeed(IN_4);
17     while(true) {
18         OnFwd(OUT_AC,50);
19         while(SensorUS(IN_4)>NEAR)
20             Off(OUT_AC);
21         // OnRev(OUT_AC,100);
22         Wait(800);
23     }
24 }
25
26 task main() {
27     sensorUltrassonico1();
28 }

```

```

1  #define NEAR 15 //cm
2
3  //usar sensor na porta 4
4  void sensorUltrassonico() {
5      SetSensorLowSpeed(IN_4);
6      while(true) {
7          OnFwd(OUT_AC,50);
8          while(SensorUS(IN_4)>NEAR);
9          Off(OUT_AC);
10         OnRev(OUT_AC,100);
11         Wait(800);
12     }
13 }

```

Além disso foi criado um arquivo *MakeFile* para definir regras e facilitar compilação dos códigos realizados.

```

1  download:
2      nbc -d -S=usb mission1.nxc moviment.nxc
3      nbc -d -S=usb mission2.nxc moviment.nxc sensor.nxc
4
5  compile:
6      nbc -O=mission1 mission1.nxc moviment.nxc
7      nbc -O=mission2 mission2.nxc moviment.nxc
8  run:

```

9	<code>nbc -d -S=usb mission1.nxc moviment.nxc</code>
---	--

Referências