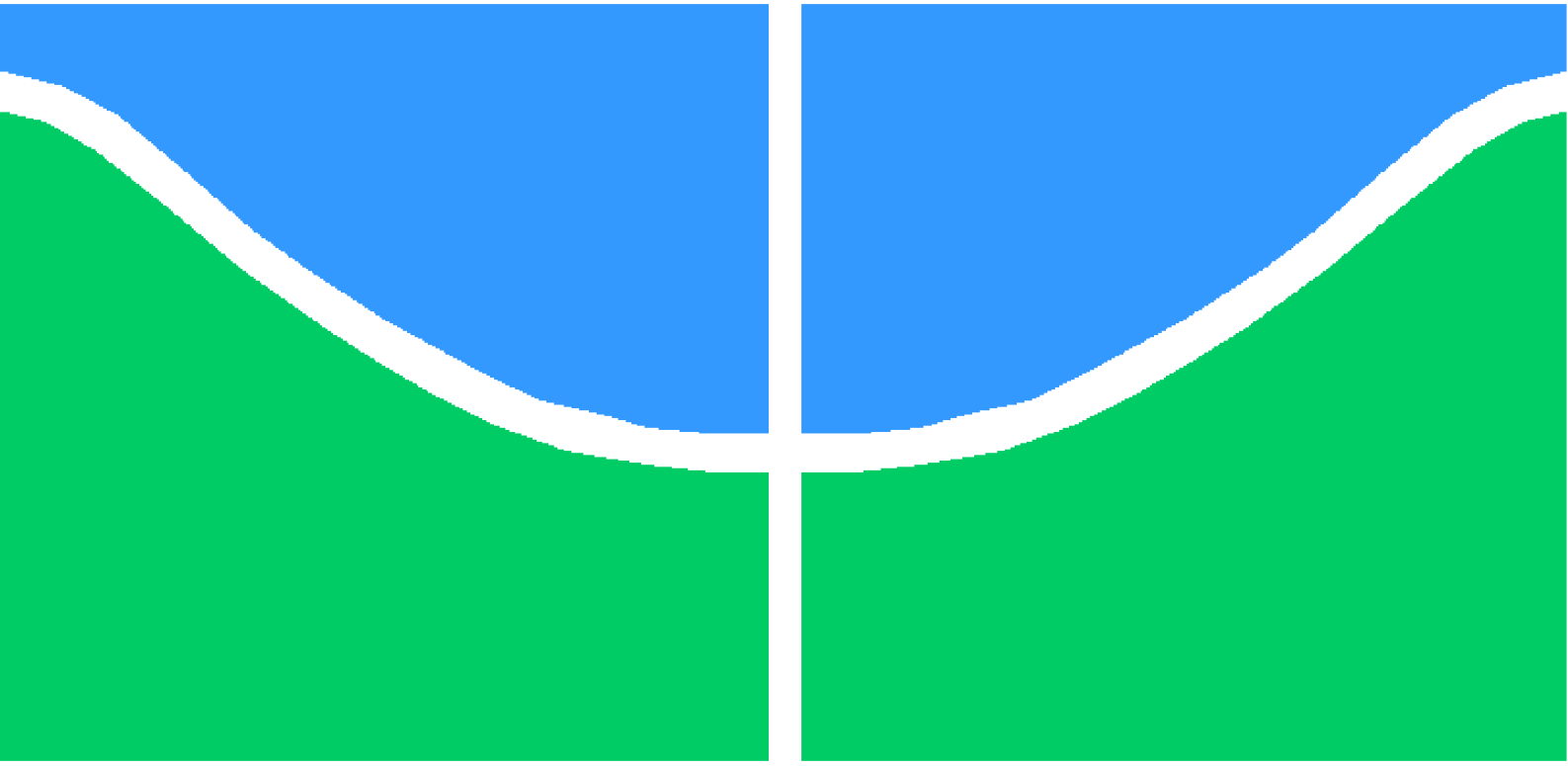

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

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 Computadores.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Brasília, DF

2015

Sumário

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 concluidas.

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
Family	O robô deve deve carregar três pessoas juntas (a famíia), até uma área colorida.	33*2	Média
Tree Branch	O robô deve derrubar o galho da árvore sem deixar que caia no poste.	30	Médio
Safe Place	O robô deve chegar com segurança à área vermelha do mapa.	25	Fácil
Uso de sensor	O robô deve usar, de maneira util, 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	Dificil
Water	O robô deve carregar a pessoa junto com a água	15	Fácil
Safety	Pessoas devem permanecer na área vermelha do mapa durante a execução das missões.	18	Média
Gerador	O robô deve levar o gerador até a área vermelha.	4	Fácil
Runway	Nada deve tocar a área de evacuação do tapete (área próxima à seta).	30	Média

5.2 Contagem dos pontos

$$T1 = 40 + 80 + 75 + 5 + 9 + 18 = 227 \tag{5.1}$$

$$T2 = 30 + 33 + 30 + 18 + 4 = 115 \tag{5.2}$$

$$T3 = 33 + 15 + 18 = 66 \tag{5.3}$$

$$TOTAL = 227 + 115 + 66 = 407 \tag{5.4}$$

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 dificil é uma tarefa comparado à quantidade de pontos que ela proporciona, e se era possível realizar tal tarefa (nem todas as tarefas dispõe dos obstáculos neste momento).

A priorização escolhida foi:

1. Caminhão e Ambulância
2. Supplies and Equipment;
3. Voltar para a base
4. Tsunami com Sensor;
5. Captura item e leva para a base;
6. Galho da Arvore;
7. Seta
8. Safe Place
9. Música
10. Runway;
11. Safety;

5.4 Justificativa

Como já citado, seguindo-se um fluxo comum de tarefas, o tempo é otimizado. Começa-se com a tarefa Caminhão e Ambulância, pois é a missão mais difícil, e, caso ela fracasse, uma nova tentativa já é feita. Em seguida, volta-se a base, e então é feito o Tsunami. O Tsunami fica perto do galho, dos itens, e da base, logo essas três coisas são feitas uma após a outra, mas a ordem não importa. Ao fim, o robô volta à base, é ajeitado, e a missão da seta é feita, em seguida a do gerador, a música é tocada, finalizando assim o desafio.

6 Solução em Software das missões

Para a codificação das missões sentiu-se a necessidade de realizar a modularização das implentações 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 andarFrenteRapido(int distancia){
12     int angulo = defineAngulo(distancia);
13     RotateMotor(OUT_AC, 100, angulo);
14 }
15
16 void andarFrente(int distancia){
17     int angulo = defineAngulo(distancia);
18     RotateMotor(OUT_AC, POTENCIA, angulo);
19 }
20
21 void andarTras(int distancia){
22     int angulo = defineAngulo(distancia);
23     RotateMotor(OUT_AC, -POTENCIA, angulo);
24 }
25
26 void andarTrasRapido(int distancia){
27     int angulo = defineAngulo(distancia);
28     RotateMotor(OUT_AC, -100, angulo);
29 }
30
31 int girar(int angulo){
32     int converte = (38.5*angulo/180);
33
34     return converte;
35 }
36
37 void virarEsquerda(int angulo){
```

```
38      int distancia = girar(angulo);
39      int anguloR = defineAngulo(distancia);
40
41      RotateMotorEx(OUT_AC, POTENCIA, anguloR, -100, true, true);
42  }
43
44  void virarDireita(int angulo){
45      int distancia = girar(angulo);
46      int anguloR = defineAngulo(distancia);
47
48      RotateMotorEx(OUT_AC, POTENCIA, anguloR, 100, true, true);
49  }
50
51  void virarDireitaPouco(int angulo){
52      int distancia = girar(angulo);
53      int anguloR = defineAngulo(distancia);
54
55      RotateMotorEx(OUT_AC, POTENCIA, anguloR, 50, true, true);
56  }
57
58  void virarEsquerdaPouco(int angulo){
59      int distancia = girar(angulo);
60      int anguloR = defineAngulo(distancia);
61
62      RotateMotorEx(OUT_AC, POTENCIA, anguloR, -50, true, true);
63  }
64
65
66  void baixarGarra(int angulo){
67      RotateMotor(OUT_B, 50, angulo);
68  }
69
70  void levantarGarra(int angulo){
71      RotateMotor(OUT_B, 100, -angulo);
72  }
```

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  #include "music.nxc"
3  #include "sensor.nxc"
```

```
4
5  task main(){
6      andarFrente(47);
7      virarDireita(90);
8      andarFrente(9);
9      virarDireita(12);
10     andarFrente(30);
11     virarEsquerdaPouco(15);
12     andarFrenteRapido(85);
13     virarEsquerdaPouco(15);
14     andarFrenteRapido(70);
15
16     //mission2 Seta
17     andarTrasRapido(70);
18     virarDireita(36);
19     andarFrente(60);
20     andarTras(20);
21     virarDireita(22);
22     andarFrente(55);
23     PlayMario();
24
25     // sensorLuz();
26 }
```

```
1  #include "moviment.nxc"
2  #include "music.nxc"
3  #include "sensor.nxc"
4
5  task main(){
6      sensorUltrassonico1(26);
7      levantarGarra(35);
8      baixarGarra(20);
9      andarTras(35);
10     //PlayMario();
11 }
```

```
1  #define NEAR 15 //cm
2  #define THRESHOLD 40
3
4  //usar sensor na porta 4
5  void sensorUltrassonico(){
6      SetSensorLowspeed(IN_4);
7      while(true){
8          OnFwd(OUT_AC,50);
```

```
9           while(SensorUS(IN_4)>NEAR);
10          Off(OUT_AC);
11          OnRev(OUT_AC,100);
12          Wait(800);
13      }
14  }
15
16  void sensorUltrassonico1(int distancia){
17      SetSensorLowspeed(IN_4);
18      while(true){
19          OnFwd(OUT_AC,76);
20          Wait(800);
21          if(SensorUS(IN_4) <= distancia){
22              Off(OUT_AC);
23              break;
24          }
25
26      }
27  }
28
29  // #define COLORSENSOR SENSOR_2 -- ou outra porta onde esteja o sensor de cor (PORTA S2
30
31  void sensorLuz(){
32      SetSensorColorRed(IN_3); // ou SetSensor(S3,SENSOR_COLORRED);
33      OnFwd(OUT_AC, 75);
34      while (true){
35          if (Sensor(IN_3) > THRESHOLD){
36              OnRev(OUT_C, 75);
37              Wait(100);
38              ClearScreen();
39              // NumOut(0, 0, COLORSENSOR);
40              until(Sensor(IN_3) <= THRESHOLD);
41              OnFwd(OUT_AC, 75);
42          }
43      }
44  }
```

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 music.nxc
4      nbc -d -S=usb mission3.nxc moviment.nxc sensor.nxc music.nxc
5
6  compile:
```

```
 7      nbc -O=mission1 mission1.nxc moviment.nxc sensor.nxc
 8      nbc -O=mission2 mission2.nxc moviment.nxc sensor.nxc
 9      nbc -O=mission2 mission2.nxc moviment.nxc sensor.nxc
10  run:
11      nbc -d -S=usb mission1.nxc moviment.nxc
```

Referências