

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	Visão geral do robô.	20
2	Vista frontal.	21
3	Vista lateral.	22
4	Vista inferior.	23
5	Base para acoplamento dos veículos.	24
6	Garra para derrubar o galho e capturar objetos.	25
7	Braço para acionamento do mecanismo das ondas.	26

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	5
2	API	6
2.1	Objetivo	6
2.2	Código NXC	6
3	Caminhão de Suplementos, Ambulância, Sinal de Evacuação e Isolamento de Construção	8
3.1	Objetivo da missão	8
3.2	Passos envolvidos	8
3.3	Pseudo código	9
3.4	Código NXC	9
4	Galho da Árvore, Animais e Equipamentos	11
4.1	Objetivo da Missão	11
4.2	Passos envolvidos	11
4.3	Pseudo-código	11
4.4	Código NXC	12
5	Tsunami	13
5.1	Objetivo da missão	13
5.2	Passos envolvidos	13
5.3	Pseudo-código	13
5.4	Código NXC	13
6	Família, Água, Segurança, Animais, Suplementos e Equipamentos, Zona de Segurança	15
6.1	Objetivo da missão	15
6.2	Passos envolvidos	15
6.3	Pseudo-código	15
6.4	Código NXC	16
7	Musica - Game of Thrones Main Theme	17
7.1	Objetivo	17
7.2	Código NXC	17
8	Robô	20
9	Garras	21

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

Lista de missões ordenadas por dificuldade, das mais fáceis para as mais difíceis, onde cada membro do grupo classificou individualmente cada missão e o resultado de todos foi mesclado nesta única tabela.

A partir daqui foi feito um cruzamento entre a pontuação e a dificuldade de cada missão para priorização das mesmas, e assim decidir a ordem em que o robô executará elas, para obter assim maior quantidade de pontos no menor espaço de tempo possível, ou seja, os 2:30 minutos.

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

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

2 API

2.1 Objetivo

O objetivo da API (Application Programming Interface) é disponibilizar um conjunto de rotinas e padrões que serão utilizados em um ou mais Softwares em que os detalhes de sua implementação não são tão importantes, mas a boa execução de seus serviços sim.

Para o caso deste projeto, uma API foi implementada visando separar as funções básicas que são mais utilizadas pelo robô para executar as missões. As rotinas identificadas foram:

- Virar – Foi observada a necessidade do robô fazer curvas durante diversas missões;
- Mover – É a operação mais básica realizada pelo robô;
- Controle de Garra – Outra função bastante utilizada em diversas missões;

Além disto, todas estas funções abstraíram características da linguagem NXC, que trabalha muito com graus, tempo (em segundos) e curvas de apenas um motor, para executar suas próprias rotinas e adaptou-as para medições mais utilizadas, como se mover em centímetros, e fazer curvas em graus (já considerando ambos os motores).

2.2 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 #define RIGHT -1
8 #define LEFT 1
9
10 #define FORWARD 1
11 #define BACKWARD -1
12
13 #define POWER_NORMAL 75
14 #define POWER_LOW 50
15 #define POWER VERY_LOW 25
16 #define POWER_HIGH 100
17
18 #define COMPLETE_ROTATION 360
19 #define HALF_ROTATION 180
```

```

20
21 sub turn(int angle, int power, int direction ) {
22     //Measured in centimeter
23     int car_arc = ( angle * PI * CAR_RADIUS ) /
24         HALF_ROTATION;
25
26     // Measured in centimeter
27     int amount_turns = (HALF_ROTATION * car_arc) /
28         (PI * WHEEL_RADIUS);
29
30     RotateMotorExPID(OUT_BC, power, amount_turns,
31                     100 * direction, true, true, 40, 40, 90);
32 }
33
34 sub move(int distance, int power, int direction) {
35     int angle = (distance * COMPLETE_ROTATION * direction) /
36         (PI * WHEEL_DIAMETER);
37
38     RotateMotorEx(OUT_BC, power, angle, 0, true, true);
39 }
40
41 sub claw_control(int angle, int power, int direction) {
42     RotateMotor(OUT_A, power, angle * direction);
43 }
```

```

1 sub turn(int angle, int power, int direction );
2
3 sub move(int distance, int power, int direction);
4
5 sub claw_control(int angle, int power, int direction);
```

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

3.1 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.)

3.2 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

3.3 Pseudo código

1. Mova em frente 38cm
2. Vire à direita 85 graus
3. Mova em frente 50cm, pegue o caminhão
4. Vire à esquerda 9 graus
5. Mova em frente 20cm, pegue a ambulância
6. Mova em frente 100cm, para chegar na área azul
7. Vire à esquerda 80 graus
8. Mova em frente 25cm
9. Mova para trás 20cm
10. Vire à esquerda 30 graus
11. Mova para trás 40cm
12. Vire à direita 35 graus
13. Mova em frente 21cm, para empurrar a placa
14. Mova para trás 30cm
15. Vire à esquerda 90 graus
16. Mova em frente 44cm
17. Mova para trás 15cm
18. Vire à esquerda 83 graus
19. Mova em frente 150cm
20. 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
4   * inside.
5   */
6  #include "theBugAPI.h"
7
8  task main() {
9
```

```

10  //Moving to Supply Truck and Ambulance.
11  move(38, POWER_NORMAL, FORWARD);
12  turn(85, POWER_NORMAL, RIGHT);
13  Wait(1);
14  move(50, POWER_NORMAL, FORWARD);
15  turn(9, POWER_LOW, LEFT);
16  Wait(1);
17  move(20, POWER_NORMAL, FORWARD);

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  //Moving to Evacuation Signal.
24  move(20, POWER_NORMAL, BACKWARD);
25  turn(30, POWER_NORMAL, LEFT);
26  Wait(1);
27  move(40, POWER_NORMAL, BACKWARD);
28  turn(35, POWER_NORMAL, RIGHT);
29  Wait(1);
30  move(21, POWER_NORMAL, FORWARD);

31  //Moving to Isolamento de Construção.
32  move(30, POWER_NORMAL, BACKWARD);
33  turn(90, POWER_NORMAL, LEFT);
34  Wait(1);
35  move(30, POWER_NORMAL, FORWARD);
36  move(14, POWER_HIGH, FORWARD);

37  //Moving to base
38  move(15, POWER_NORMAL, BACKWARD);
39  turn(83, POWER_NORMAL, LEFT);
40  Wait(1);
41  move(150, POWER_HIGH, FORWARD);
42  }

```

4 Galho da Árvore, Animais e Equipamentos

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;
- 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).

4.2 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

4.3 Pseudo-código

1. Mova em frente 70 cm
2. Levante a garra 100 graus
3. Desça a garra 100 graus
4. Mova para trás 30cm
5. Vire à direita 85 graus
6. Mova para trás 22cm
7. Vire à esquerda 90 graus
8. Mova em frente 39cm
9. Levante a garra 70 graus
10. Mova para trás 70cm
11. Encerre o programa

4.4 Código NXC

```
1  /*
2  * Initial Position:
3  * Right side of the "claw", aligned at the end of the "M" by
4  * left-inside.
5  */
6 #include "theBugAPI.h"
7
8 task main() {
9
10    //Moving to the tree.
11    move(70, POWER_NORMAL, FORWARD);
12    claw_control(10, POWER_HIGH, BACKWARD);
13    claw_control(90, POWER_HIGH, BACKWARD);
14    claw_control(100, POWER_HIGH, FORWARD);
15
16    //Moving to catch the animals.
17    move(30, POWER_NORMAL, BACKWARD);
18    turn(85, POWER_NORMAL, RIGHT);
19    Wait(1);
20    move(22, POWER_NORMAL, BACKWARD);
21    turn(90, POWER_NORMAL, LEFT);
22    Wait(1);
23    move(39, POWER_NORMAL, FORWARD);
24    claw_control(70, POWER_NORMAL, BACKWARD);
25
26    //Moving to base.
27    move(70, POWER_HIGH, BACKWARD);
28 }
```

5 Tsunami

5.1 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.

5.2 Passos envolvidos

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

5.3 Pseudo-código

1. Ligue o sensor Ultrasônico
2. Mova em frente até que o sensor leia uma distância de 23cm ou menos
3. Levante a garra 60 graus
4. Mova para trás 70cm
5. Encerre o programa

5.4 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 }
```

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

6.1 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.

6.2 Passos envolvidos

Sair da base

Pegar a pessoa ao lado da casa

Ir para a região vermelha

6.3 Pseudo-código

1. Mova em frente 43cm
2. Vire à direita 85 graus
3. Mova em frente 67cm
4. Vire à esquerda 78 graus
5. Mova em frente 22cm
6. Levante a garra 55 graus
7. Mova para trás 15cm
8. Vire à direita 85 graus
9. Mova em frente 80cm
10. Vire à direita 44cm
11. Ligue o sensor RGB
12. Mova em frente até o sensor detectar a cor vermelha
13. Encerre o programa

6.4 Código NXC

```
1  /*
2  * Initial Position:
3  * Right side of the "claw", aligned at the end of the "M" by
4  * left-inside.
5  */
6  #include "theBugAPI.h"
7
8  #define COLORSENSOR_SENSOR_2
9  #define RED 5
10
11 task main() {
12
13     //Moving to the father
14     move(43, POWER_HIGH, FORWARD);
15     turn(85, POWER_HIGH, RIGHT);
16     Wait(1);
17     move(67, POWER_HIGH, FORWARD);
18     turn(78, POWER_LOW, LEFT);
19     Wait(1);
20     move(22, POWER_HIGH, FORWARD);
21     claw_control(55, POWER_NORMAL, BACKWARD);
22
23     //Moving to the red zone
24     move(15, POWER_HIGH, BACKWARD);
25     turn(85, POWER_HIGH, RIGHT);
26     Wait(1);
27     move(80, POWER_HIGH, FORWARD);
28     turn(44, POWER_HIGH, RIGHT);
29     Wait(1);
30
31     //Turn on the RGB sensor
32     SetSensorColorFull(S2);
33
34     while(Sensor(S2) != RED) {
35         OnFwd(OUT_BC, POWER_NORMAL);
36     }
37
38     PlaySound(GOT_RAINS_OF_CASTAMERE);
39 }
```

7 Musica - Game of Thrones Main Theme

7.1 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.

7.2 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);
25     Wait(COLCHEIA);
```

```

26     PlayToneEx(440, COLCHEIA, VOL, FALSE);
27     Wait(COLCHEIA);
28     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE);
29     Wait(SEMICOLCHEIA);
30     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE);
31     Wait(SEMICOLCHEIA);
32     PlayToneEx(659, COLCHEIA, VOL, FALSE);
33     Wait(COLCHEIA);
34     PlayToneEx(440, COLCHEIA, VOL, FALSE);
35     Wait(COLCHEIA);
36     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE);
37     Wait(SEMICOLCHEIA);
38     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE);
39     Wait(SEMICOLCHEIA);
40     PlayToneEx(659, COLCHEIA, VOL, FALSE);
41     Wait(COLCHEIA);
42     PlayToneEx(440, COLCHEIA, VOL, FALSE);
43     Wait(COLCHEIA);
44     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE);
45     Wait(SEMICOLCHEIA);
46     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE);
47     Wait(SEMICOLCHEIA);
48     PlayToneEx(659, COLCHEIA, VOL, FALSE);
49     Wait(COLCHEIA);
50     PlayToneEx(440, COLCHEIA, VOL, FALSE);
51     Wait(COLCHEIA);
52     PlayToneEx(523, SEMICOLCHEIA, VOL, FALSE);
53     Wait(SEMICOLCHEIA);
54     PlayToneEx(587, SEMICOLCHEIA, VOL, FALSE);
55     Wait(SEMICOLCHEIA);
56 }
57
58 task main() {
59
60     // Main theme of game of thrones.
61     PlayToneEx(392, SEMINIMA_PONTUADA, VOL, FALSE);
62     Wait(SEMINIMA_PONTUADA);
63     PlayToneEx(264, SEMINIMA_PONTUADA, VOL, FALSE);
64     Wait(SEMINIMA_PONTUADA);
65     PlayToneEx(311, SEMICOLCHEIA, VOL, FALSE);
66     Wait(SEMICOLCHEIA);
67     PlayToneEx(349, SEMICOLCHEIA, VOL, FALSE);
68     Wait(SEMICOLCHEIA);
69     PlayToneEx(392, SEMINIMA, VOL, FALSE);
70     Wait(SEMINIMA);
71     PlayToneEx(262, SEMINIMA, VOL, FALSE);

```

```
72     Wait(SEMINIMA);
73     PlayToneEx(311, SEMICOLCHEIA, VOL, FALSE);
74     Wait(SEMICOLCHEIA);
75     PlayToneEx(349, SEMICOLCHEIA, VOL, FALSE);
76     Wait(SEMICOLCHEIA);
77     PlayToneEx(294, SEMIBREVE_PONTUADA, VOL, FALSE);
78     Wait(SEMIBREVE_PONTUADA);
79
80     background_strings();
81 }
```

8 Robô

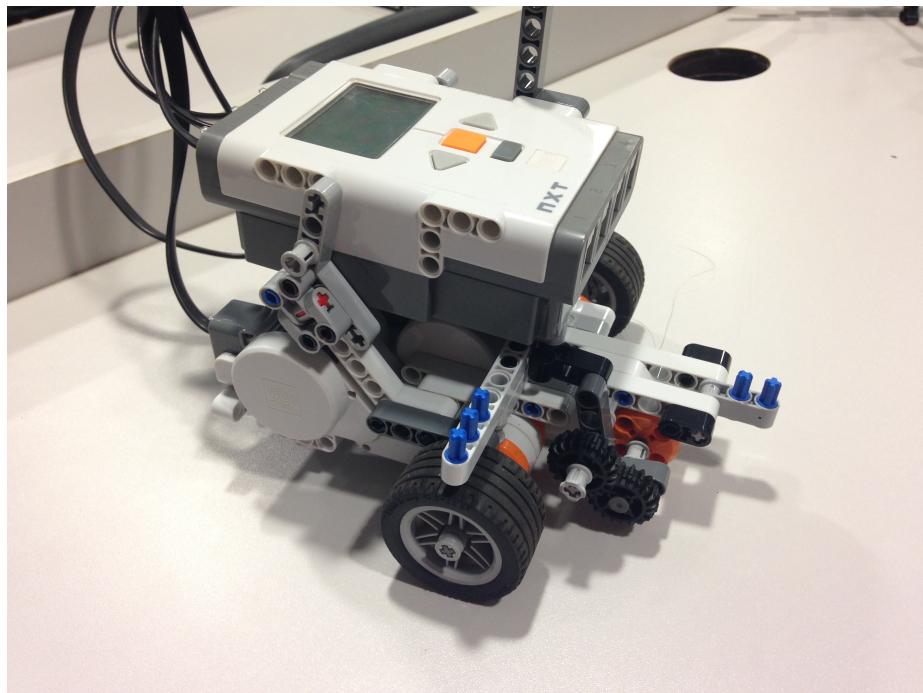


Figura 1: Visão geral do robô.

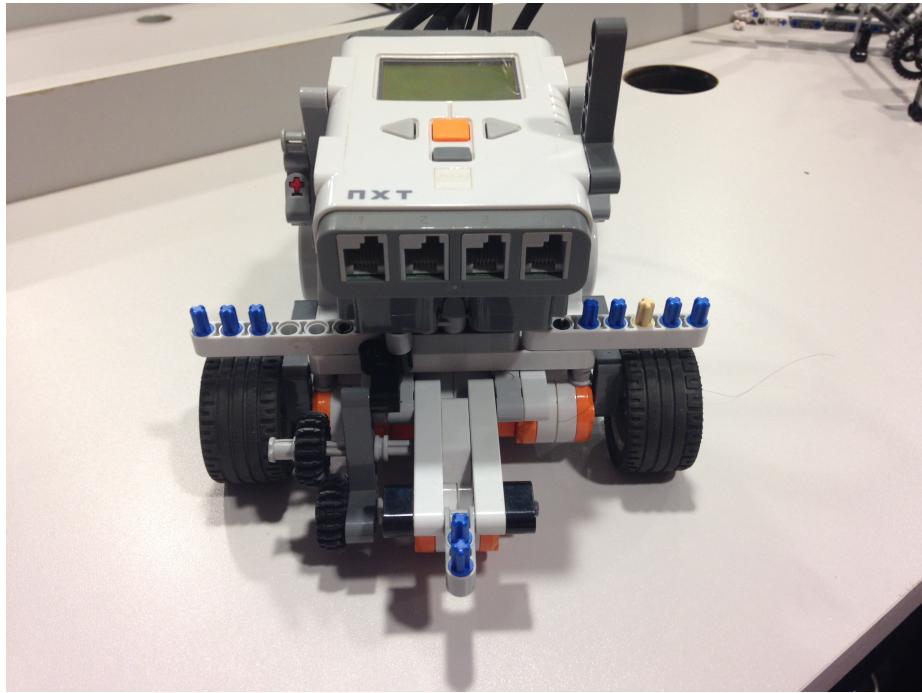


Figura 2: Vista frontal.

9 Gerras

Para que as missões fossem efetivamente cumpridas, três garras removíveis foram montadas. O objetivo foi trocar de garra para um grupo de missões específicas, afim de que as mesmas pudessem ser realizadas da melhor forma possível.

A Figura 5 mostra a primeira garra que foi construída. O objetivo desta é agrupar a ambulância e o caminhão de suplementos dentro da garra e levá-los até a área pretendida.

A Figura 6 mostra a garra construída com o objetivo de se levantar para derrubar o galho da árvore e capturar objetos para levá-los em áreas distintas. Ela também possui um sensor RGB com finalidade de perceber as cores do tapete enviando um sinal para o microcontrolador que irá parar o robô caso receba um sinal de cor vermelha.

A Figura 7 mostra a garra que tem como objetivo único de se levantar para acionar o mecanismo que derruba as “ondas”.

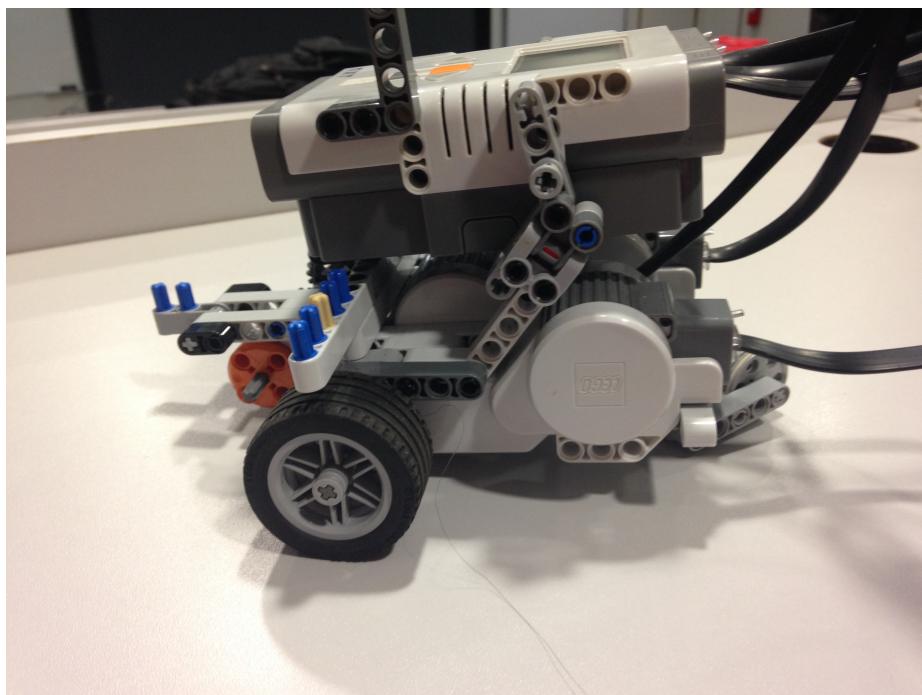


Figura 3: Vista lateral.

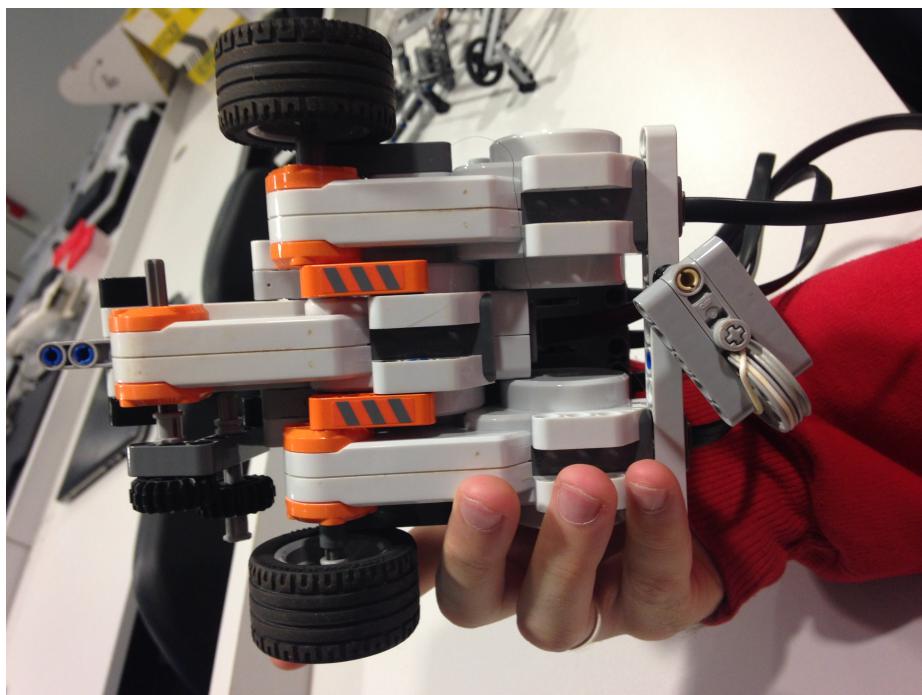


Figura 4: Vista inferior.

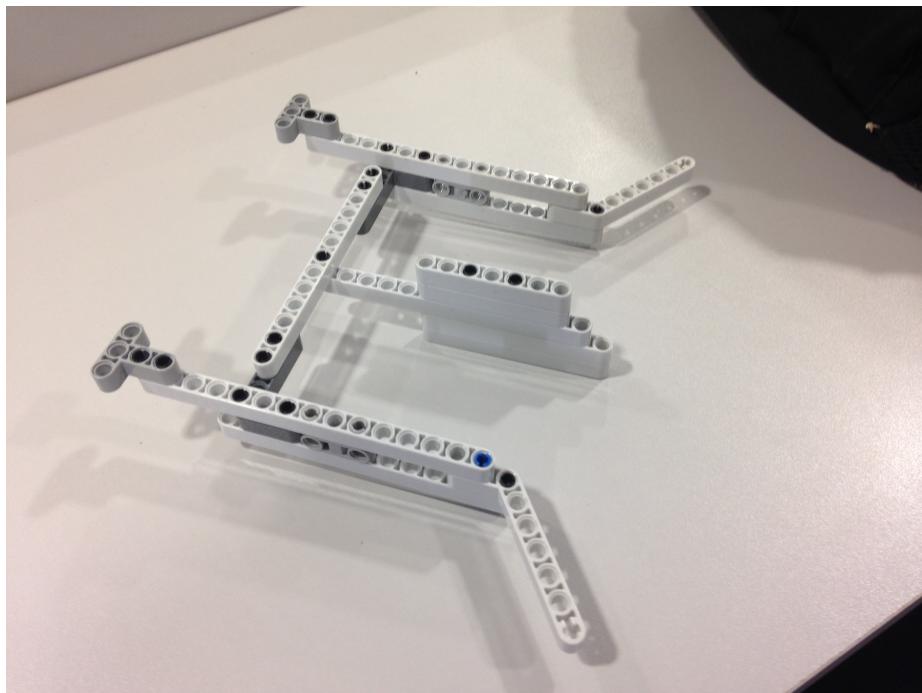


Figura 5: Base para acoplamento dos veículos.



Figura 6: Garra para derrubar o galho e capturar objetos.



Figura 7: Braço para acionamento do mecanismo das ondas.