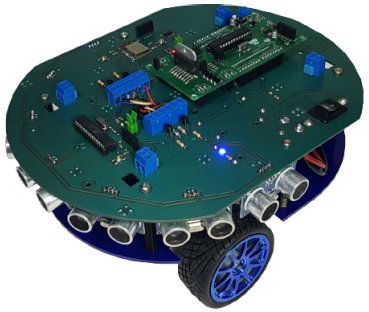
Edubot IDE 1.0: Folha de comandos

Carlos Eduardo Pedroso de Oliveira
Marcos Rodrigues Vizzotto
Maik Basso
Renato Ventura Bayan Henriques

Outubro 2020



Contents

1 Visão Geral

Edubot IDE

O ambiente de desenvolvimento integrado para o Edubot permite o desenvolvimento de aplicações para o robô. A IDE possui um *joystick*, que possibilita o controle manual do Edubot e um simulador 2D para testes. Este documento apresenta um guia das funções disponibilizadas pela biblioteca EdubotLib, especificações do robô e guia rápido para uso da IDE. Todos exemplos apresentados disponíveis através do link

2 Comandos EdubotLib

Movimentação

Movimentar

`bool move(double velocity)` ..... mov. linear **[velocity: -1 a 1]**
`bool rotate(double angle)` ..... mov. rotação **[angle: -180 a 180]**

Parar

`bool stop()` ..... freia o robô
`bool neutral()` ..... motores não atuados

Rotação
Direita: +0 a +180°
Esquerda: -0 a -180°

Sensores

Argumento `size_t id` deve ser um inteiro positivo representando a identificação do sensor. Consultar Figura 2 para identificação dos sonares.

Sensores

`double getSonar(size_t id)` ..... leitura do sonar (m) **[id: 0 a 6]**
`bool getBumper(size_t id)` ..... verdadeiro: contato no bumper **[id: 0 a 3]**
`int getEncoderCountLeft()` ..... contagem encoder esq.
`int getEncoderCountRight()` ..... contagem encoder dir.
`int getEncoderCountDT()` ..... tempo (ms) de loop entre as contagens
`double getBatteryCellVoltage(size_t id)` ..... tensão (V) das células da bateria **[id: 0 a 3]**

Posição

`double getX()` ..... posição x (m)
`double getY()` ..... posição y (m)
`double getTheta()` ..... ângulo (°)

Conexão e outros

Conexão

`bool isConnected()` ..... verdadeiro: robô conectado
`bool connect()` ..... conecta ao robô
`bool disconnect()` ..... desconecta do robô

Sleep

`void sleepMilliseconds(int ms)` ..... suspende execução por ms

3 Programação

Exemplo movimentos

```cpp
/*
 * EDUBOT Exemplo 1
 * @Author: Carlos Eduardo
 * Esse exemplo demonstra as funcoes de movimentacao do robo:
 * move(double velocity) -> movimento linear
 * stop() -> freia os motores
 * rotate(double angle) -> movimento rotacional
 * neutral() -> coloca os motores em neutro
 */

#include <iostream>
#include "libs/EdubotLib.hpp"

int main(){
    /*
     * vamos criar um ponteiro para um objeto da classe edubot
     * esse elemento nos permitira acessar todas as funcoes que sao
     * fornecidas pela biblioteca
     */
    EdubotLib *edubotLib = new EdubotLib();
       /*
      * primeiro precisamos nos conectar ao robo
      * a funcao connect retorna verdadeiro em caso de sucesso
       */
    if(edubotLib->connect()){
        /*
         * Funcao move > move(double velocity)
         * Deve receber um valor double entre (-1.0, 1.0)
         * Valores positivos move para frente negativos para tras
        */
        edubotLib->move(0.5);
        /*
         * Para permitir que o robo se movimente por um determinado tempo
         * se utiliza da funcao sleepMilliseconds(int ms), que suspende a
         * ezecucao do programa pelo valor de milissegundos passado em seu
         * argumento
         */
        edubotLib->sleepMilliseconds(2000);
        /*
         * Funcao rotate > rotate(double angle)
         * Deve receber um angulo em graus entre (-180, 180)
         * Valores positivos rotaciona para direita, negativos para a esquerda
        */
        edubotLib->rotate(90);
        // Mais uma vez aguarda um periodo para o robo poder rotacionar
        edubotLib->sleepMilliseconds(2000);
        // Agora movimenta o robo para tras
        edubotLib->move(-0.5);
        edubotLib->sleepMilliseconds(2000);
        /*
         * Funcao stop
         * Utilizada para frear os motores
         */
        edubotLib->stop();
        // Aguarda 1 segundo para garantir que o robo esta parado
        edubotLib->sleepMilliseconds(1000);
        /*
         * Funcao neutral
         * coloca os motores em estado neutro
        */
        edubotLib->neutral();
        edubotLib->sleepMilliseconds(500);
        /*
         * Funcao disconnect
         * desconecta o programa do robo
         * SEMPRE deve ser utilizada
        */
        edubotLib->disconnect();
    }
    else{
        // Caso nao seja possivel conectar ao robo, enviar um aviso na tela da ide
        std::cout << "Could not connect on robot!" << std::endl;
    }
    return 0;
}
```

Trajetoria quadrado

```cpp
#include <iostream>
#include "libs/EdubotLib.hpp"

int main(){
    EdubotLib *edubotLib = new EdubotLib();
    //try to connect on robot
    if(edubotLib->connect()){
        for(;;){
            edubotLib->move(0.5);
            edubotLib->sleepMilliseconds(1000);
            edubotLib->rotate(90);
            edubotLib->sleepMilliseconds(1000);
        }
        edubotLib->disconnect();
    }
    else{
        std::cout << "Could not connect on robot!" << std::endl;
    }
    return 0;
}
```

Leitura Sensores - Parte 1

```cpp
/*
 * EDUBOT Exemplo 3
 * @Author: Carlos Eduardo
 * Nesse exemplo iremos obter leituras de alguns dos sensores do robo
 * As leituras sao apresentadas no console. Funcoes utilizadas:
 *  double getSonar(size_t id) -> leitura dos sonares em metros
 *  bool getBumper(size_t id) -> leitura dos bumbers
 *  int getEncoderCountLeft() -> leitura do encoder da roda esquerda
 *  int getEncoderCountRight() -> leitura do encoder da roda direita
 *  int getEncoderCountDT() -> tempo de loop dos encoders
 *  double getX() -> Posicao X do robo com relacao ao sist. de coordenadas inicial
 *  double getY() -> Posicao Y do robo com relacao ao sist. de coordenadas inicial
 *  double getTheta() -> Angulo de giro do robo com relacao ao sist. de coordenadas inicial
 *  double getBatteryCellVoltage(size_t id) -> Tensao nas celulas da bateria
*/
#include <iostream>
#include "libs/EdubotLib.hpp"

#define N_SONARES 7
#define N_BUMPERS 4
#define N_CELULAS 3

using namespace std;

int main(){

        EdubotLib *edubotLib = new EdubotLib();
        double posX;
        double sonares[N_SONARES];

        if(edubotLib->connect()){

                /*
                 * Vamos fazer a leitura ciclica dos sensores a cada 1 segundo aproximadamente
                 */
                while(edubotLib->isConnected()){
                        cout<<"---"<<endl;

                        /*
                         * Primeiro a leitura dos 7 sonares a funcao
                         * getSonar(size_t id) deve receber como argumento um valor inteiro positivo (0 ate 6)
                         * retorna um valor do tipo double com a distancia de um obstaculo em metros
                         * a distancia maxima de deteccao: 2m
                         */

                        for(int i=0; i<N_SONARES; i++){
                                sonares[i]=edubotLib->getSonar(i);
                                cout<<"Sonar "<<i<<": "<<sonares[i]<<"m, ";
                        }
                        cout<<endl;
```

Leitura Sensores - Parte 2

```
                        /*
                         * A leitura dos bumpers realizada com a funcao
                         * getBumper(size_t id) recebe como argumento um valor inteiro positivo (0 ate 3)
                         * retorna um valor booleano (0 - falso, 1 - verdadeiro) indicando se
                         * houve ou no contato do robo com algum obstaculo
                         */
                        for(int i=0; i<N_BUMPERS; i++){
                                cout<<"Bumper "<<i<<": "<<edubotLib->getBumper(i);
                        }
                        cout<<endl;

                        /*
                         * As funcoes getX(), getY() e getTheta() nao representam a leitura
                         * direta de um sensor, mas sim os valores calculados pelo robo para
                         * sua posicao no plano. Retornam valor do tipo double.
                         * A posicao eh fornecida com respeito a posicao inicial
                         */
                        cout<<"X: "<<edubotLib->getX();
                        cout<<", Y: "<<edubotLib->getY();
                        cout<<", Theta: "<<edubotLib->getTheta()<<endl;

                        // Poderia armazenar a posicao em uma variavel
                        posX = edubotLib->getX();

                        /*
                         * A medida das tensoes nas celulas da bateria
                         * getBatteryCellVoltage(size_t id) recebe como argumento um inteiro positivo (0 ate 2)
                         * retorna um valor double com o nivel de tensao em cada uma das 3 celulas deve-se monitorar
                         * para que os valores nao fiquem abaixo de 3V
                         */
                         for(int i=0; i<N_CELULAS; i++){
                                 cout<<"Celula "<<i<<": "<<edubotLib->getBatteryCellVoltage(i)<<", ";
                         }
                        cout<<endl;

                        /*
                         * Leituras de pulso dos encoders
                         * getEncoderCountLeft() e getEncoderCountRight() retornam um inteiro
                         * com a contagem de pulsos dos encoders
                         * getEncoderCountDT() retorna um valor inteiro com o tempo de loop dos encoders
                         * em milissegundos
                         * Com esses valores pode-se fazer a propria implementacao das funcoes que
                         * informam a posicao e orientacao do robo
                         */
                        cout << "Encoder Esq: " << edubotLib->getEncoderCountLeft() << ", ";
                        cout << "Encoder Dir: " << edubotLib->getEncoderCountRight() << ", ";
                        cout << "Dt: " << edubotLib->getEncoderCountDT() << endl;


                        edubotLib->sleepMilliseconds(1000);
                }

                edubotLib->disconnect();
        }
        else{
                std::cout << "Could not connect on robot!" << std::endl;
        }

        return 0;
}
```

Tomada de decisões

```
/*
 * EDUBOT Exemplo 4
 * @Author: Carlos Eduardo
 * Nesse exemplo o robo utiliza dados dos sensores
 * para tomar decisoes
 * Edubot move para frente enquanto nao encontra obstaculos
 * quando detecta algo a menos de 0.2 metros o robo gira e
 * continua se movendo para frente
 *
 */

#include <iostream>
#include "libs/EdubotLib.hpp"

using namespace std;

int main(){

        EdubotLib *edubotLib = new EdubotLib();

        //try to connect on robot
        if(edubotLib->connect()){
                edubotLib->sleepMilliseconds(200);
                while(edubotLib->isConnected()){
                        edubotLib->move(0.3);
                        do{

                        }while(edubotLib->getSonar(3)>0.2);
                        edubotLib->rotate(90);
                        edubotLib->sleepMilliseconds(1500);
                }

                edubotLib->disconnect();
        }
        else{
                std::cout << "Could not connect on robot!" << std::endl;
        }

        return 0;
}
```

4 Simulação

Utilizando a IDE

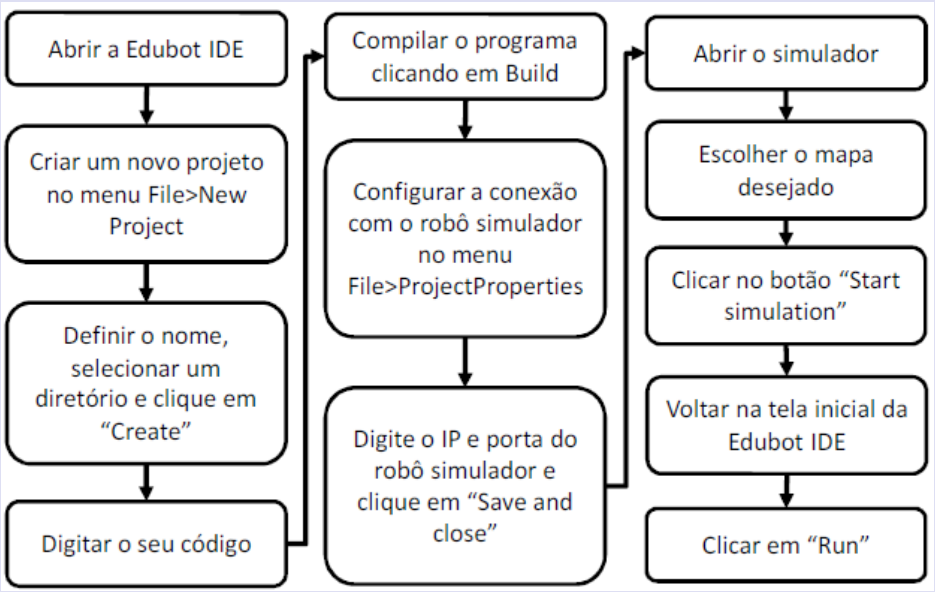


Figure 1: Sequência de passos para utilização do simulador

5 Especificações do Robô

Orientação dos Sonares

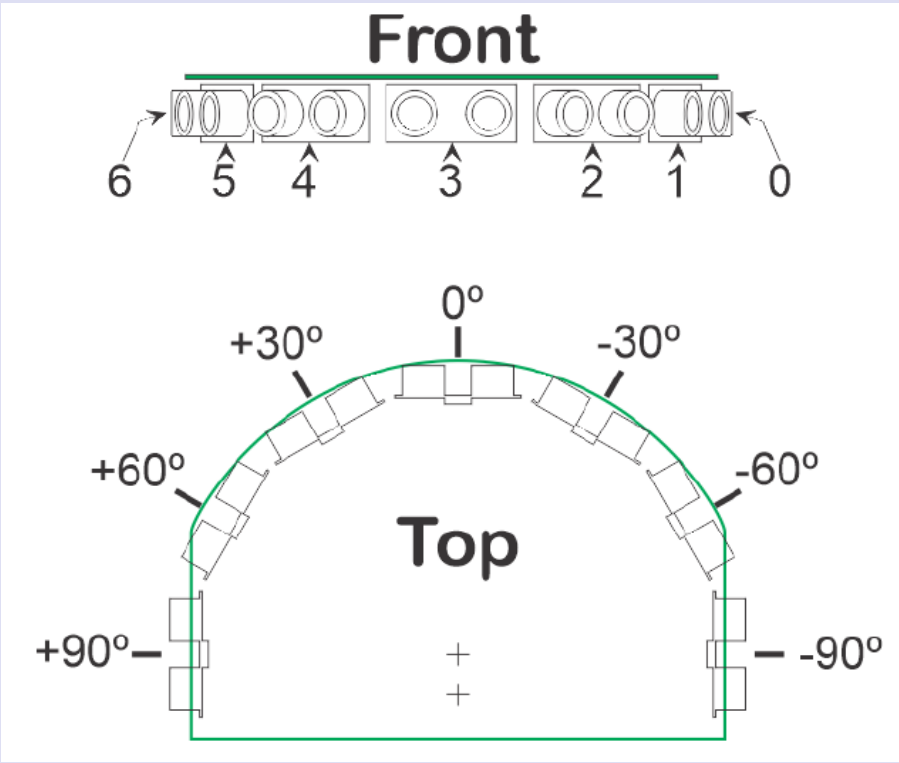


Figure 2: Orientação dos sonares