Relatório Edubot ECP

Programação do Edubot para sair de um Labirinto

Alunos:
Antonio de Castilhos Piccolo - 00590883
Arthur Velho Dohnert - 00587540
Eduardo Henrique Drumm Menezes - 00588224
Gabriel Lemos Simão de Albuquerque - 00590032
Lucas Steimetz - 00589697
Thomas Henrique Schmitz - 00588622

ENG10031 - Introdução à Engenharia de Computação
Turma U

Universidade Federal do Rio Grande do Sul
Escola de Engenharia

Porto Alegre
Brasil

20 de Agosto de 2024

Resumo

O presente relatório descreve uma aula prática de Introdução à Engenharia de Computação, em que os alunos da disciplina precisaram programar um robô chamado Edubot para que ele saísse de um labirinto. Para isso, foi utilizada a linguagem C++ e o ambiente de desenvolvimento próprio do Edubot, integrado com um simulador. Este relatório explora a metodologia aplicada pelos alunos, bem como os resultados alcançados e a conclusão elaborada, ressaltando a relevância da atividade.

Conteúdo

1 Introdução

A engenharia de computação é uma área multidisciplinar que integra elementos da ciência da computação e da engenharia elétrica para projetar sistemas computacionais e dispositivos eletrônicos, como sistemas embarcados.

O presente relatório descreve a nossa experiência em aula na programação do robô Edubot. Esta atividade proporcionou aos participantes a oportunidade de aplicar na prática os diversos conhecimentos obtidos em aula.

Durante a aula, os alunos tiveram que programar o Edubot para que ele conseguisse sair de diversos labirintos sozinho. Ao longo deste relatório serão apresentados os métodos utilizados para a programação do Edubot, assim como os resultados obtidos tanto na aplicação real, quanto no simulador.

2 Materiais utilizados

Para a programação e simulação do robô Edubot, foram utilizados os seguintes materiais e recursos:

- Robô Edubot: O Edubot é um robô educacional projetado para auxiliar no aprendizado de programação e robótica. Equipado com sensores, motores e uma placa controladora, ele permite a execução de diferentes tarefas programáveis.
- IDE Edubot: A Interface de Desenvolvimento Integrado (IDE) específica do Edubot foi utilizada para escrever, compilar e fazer upload dos códigos em C++ para o robô. Esta IDE é projetada para ser intuitiva, facilitando o desenvolvimento de programas e a integração com o simulador.
- Simulador Edubot: O simulador integrado da IDE Edubot foi utilizado para testar e depurar o código antes de implementá-lo no robô físico. O simulador permite emular o comportamento do Edubot em diferentes labirintos, proporcionando um ambiente seguro e controlado para experimentação.
- Linguagem de Programação C++: O robô Edubot foi programado utilizando a linguagem C++, que oferece a flexibilidade e o controle necessários para desenvolver algoritmos eficientes para o robô.
- Computador Pessoal: Um computador pessoal foi utilizado para acessar a IDE e o simulador do Edubot, além de servir como a plataforma de desenvolvimento para escrever e testar o código.

Esses materiais foram essenciais para a programação e simulação do Edubot, permitindo aos alunos explorar conceitos de robótica e programação de maneira prática e interativa.

3 Metodologia Utilizada

A metodologia empregada para o desenvolvimento e programação do robô Edubot envolveu uma abordagem exploratória e prática, que incentivou o autoaprendizado e a resolução autônoma de problemas pelos alunos. O processo foi dividido em várias etapas principais:

3.1 Familiarização com a IDE e o Simulador Edubot

A primeira etapa do projeto consistiu em uma imersão completa na Interface de Desenvolvimento Integrado (IDE) e no simulador específicos do Edubot. Os alunos começaram explorando a interface da IDE, que oferece diversas ferramentas e funcionalidades para a programação do robô em C++. Este processo incluiu a compreensão de como criar, compilar e carregar códigos no Edubot, além de aprender a utilizar as bibliotecas disponíveis que facilitam a interação com os componentes do robô, como sensores e atuadores. Durante essa fase, os alunos também navegaram pela documentação e tutoriais fornecidos, buscando entender as possibilidades oferecidas pela plataforma.

Paralelamente, os alunos dedicaram tempo para se acostumar com o simulador Edubot, uma ferramenta crucial que permitiu a experimentação em um ambiente virtual. O simulador reproduz fielmente o comportamento do robô em diferentes cenários, o que foi fundamental para testar códigos de forma segura antes de implementá-los no robô físico. A exploração inicial do simulador envolveu a configuração de ambientes de teste e a execução de comandos básicos, permitindo que os alunos ganhassem confiança no uso das ferramentas antes de avançar para etapas mais complexas do projeto.

3.2 Estudo Independente dos Conceitos de Programação

Dada a natureza do projeto, os alunos foram incentivados a realizar um estudo aprofundado dos conceitos de programação em C++, uma linguagem poderosa e amplamente utilizada em sistemas embarcados e robótica. Esta fase do projeto exigiu que os alunos revisitassem conceitos fundamentais, como estruturas de controle (condicionais e loops), manipulação de variáveis, funções, e o uso de arrays e ponteiros, que são essenciais para gerenciar os dados e controlar o fluxo do programa de maneira eficiente.

Além disso, os alunos precisaram se familiarizar com aspectos específicos da programação de microcontroladores, como o gerenciamento de entradas e saídas digitais, a leitura de sensores, e o controle de motores e outros atuadores. Para isso, os alunos buscaram referências em manuais, tutoriais online, e exemplos de código fornecidos pela comunidade, adaptando-os às necessidades do projeto. Esse estudo independente não apenas fortaleceu as habilidades de programação dos alunos, mas também lhes proporcionou uma compreensão mais profunda de como os conceitos teóricos são aplicados na prática.

3.3 Desenvolvimento Iterativo do Código

A programação do Edubot foi abordada de maneira iterativa, com os alunos adotando um ciclo contínuo de desenvolvimento, teste e refinamento do código. Inicialmente, foram escritos pequenos blocos de código que abordavam funções específicas do robô, como a movimentação básica, a leitura de sensores ou a resposta a comandos. Cada um desses blocos foi testado individualmente no simulador, permitindo a identificação de problemas e a realização de ajustes necessários antes de integrar todas as funcionalidades.

À medida que o projeto avançava, os alunos começaram a integrar esses blocos de código em um programa mais completo, que coordenava todas as funções do robô. Durante cada iteração, novos elementos eram adicionados, e o código existente era revisado para garantir que todas as partes funcionassem harmoniosamente. Esse processo iterativo de desenvolvimento foi crucial para a construção de um código robusto e eficiente, minimizando a ocorrência de erros e facilitando a depuração em etapas posteriores do projeto.

3.4 Testes e Depuração no Simulador

Antes de transferir o código para o robô físico, os alunos dedicaram tempo significativo à fase de testes e depuração utilizando o simulador Edubot. Esta etapa foi essencial para garantir que o programa funcionasse conforme o esperado em uma variedade de condições, permitindo identificar e corrigir erros sem o risco de danificar o hardware do robô. O simulador ofereceu um ambiente controlado onde os alunos puderam simular diferentes cenários, como obstáculos no caminho do robô ou variações nas condições de iluminação, para verificar a resposta do programa a essas situações.

Durante a depuração, os alunos utilizaram ferramentas de diagnóstico disponíveis no simulador para monitorar o comportamento do código em tempo real, identificando possíveis falhas lógicas ou bugs que poderiam comprometer o funcionamento do robô. Esse processo permitiu que os alunos refinassem o código, ajustando parâmetros e corrigindo erros antes de realizar o teste final no robô físico. A utilização intensiva do simulador não só acelerou o processo de desenvolvimento, mas também aumentou a confiança dos alunos na robustez e precisão do código desenvolvido.

3.5 Implementação no Robô Físico

Após validar o código no simulador, os alunos procederam com a transferência do programa final para o robô Edubot, marcando a fase de implementação no hardware real. Esse processo envolveu o upload do código compilado diretamente para a memória do microcontrolador do Edubot, utilizando a IDE. Uma vez carregado, o robô foi desconectado do ambiente de desenvolvimento e colocado em um cenário físico onde foi submetido a testes práticos para garantir que todas as funções programadas se comportassem conforme o esperado.

Os testes no robô físico revelaram a eficácia do código, mas também apresentaram desafios adicionais, como a calibração de sensores e ajustes finos na resposta dos atuadores, que não eram totalmente previsíveis no simulador. Esses desafios exigiram que os alunos fizessem pequenos ajustes no código diretamente no ambiente real, refinando o desempenho do robô até alcançar os resultados desejados. A implementação final no robô físico representou a culminação do processo de desenvolvimento, demonstrando a capacidade dos alunos de transformar conceitos teóricos em soluções práticas e funcionais.

Essa abordagem permitiu que os alunos desenvolvessem habilidades práticas de programação e resolução de problemas, ao mesmo tempo em que se adaptavam às necessidades do projeto de forma independente e proativa.

4 Resultados

Durante o desenvolvimento e implementação do projeto de programação do robô Edubot, foram observados desafios significativos que impactaram o sucesso inicial da execução do código, mas que, posteriormente, levaram a um aprendizado valioso e à solução dos problemas.

4.1 Primeira Tentativa

Na primeira tentativa de aplicar o código desenvolvido no robô físico, o Edubot não funcionou corretamente na vida real. Embora o código tenha sido testado extensivamente no simulador Edubot, o teste no ambiente físico não deu certo. O principal problema identificado era a leitura dos bumpers. Essa experiência inicial destacou a importância de considerar as diferenças entre o ambiente de simulação e o mundo real, onde variáveis como imprecisões mecânicas podem afetar o desempenho.

Após uma análise cuidadosa dos erros, foi necessário revisar o código e realizar ajustes específicos para corrigir as falhas encontradas. Foram realizadas depurações detalhadas para identificar qualquer outra falha que pudesse estar comprometendo o funcionamento do robô.

Um dos principais problemas que identificamos no robô foi a presença de um loop no código que, em certas situações, fazia com que o Edubot entrasse em um ciclo infinito do qual ele não conseguia sair. Isso impedia o robô de seguir corretamente as instruções e realizar as tarefas programadas. Além disso, percebemos que havia muitos comandos de espera (sleepMilliseconds()) ao longo do código, o que causava atrasos significativos na execução das ações. Esses atrasos faziam com que a rotina de colisão fosse acionada com pouca frequência, deixando o robô mais vulnerável a colisões que ele deveria evitar.

4.2 Segunda Tentativa

Na segunda aula, com o código revisado e as correções implementadas, o robô Edubot foi novamente submetido a testes no ambiente físico. Desta vez, o código funcionou conforme o esperado, demonstrando um desempenho consistente e preciso em todas as tarefas programadas. O Edubot foi capaz de interagir com seu ambiente, processar as informações recebidas dos sensores e executar as ações programadas de maneira eficiente. Esse sucesso foi um reflexo direto das melhorias feitas com base nos aprendizados da primeira tentativa, evidenciando a eficácia do processo de depuração e ajuste.

O resultado final foi um robô capaz de cumprir suas funções conforme especificado no projeto inicial. O sucesso da segunda tentativa reforçou a importância de uma abordagem iterativa no desenvolvimento de sistemas embarcados, onde o teste contínuo e a adaptação são essenciais para alcançar um funcionamento ideal.

Portanto, aqui abaixo segue o código final do grupo na programação do Edubot, com cada instrução devidamente comentada com sua funcionalidade.

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

#define MIN_DIST 0.07
// distancia minima frontal
#define SPEED 0.2
// velocidade
#define TIME 2000
// tempo de movimento em milissegundos


#define MIN_DIST_SIDE (3 * MIN_DIST)
// distancia minima lateral, 3x a distancia frontal

EdubotLib *edubot = new EdubotLib();
// criando instancia do edubot

// rotina pra lidar com colisao
bool colisionRoutine() {
    // verifica se algum dos bumpers do edubot foi ativado
    if (edubot->getBumper(0) || edubot->getBumper(1)
        || edubot->getBumper(2) || edubot->getBumper(3)) {
        float speed = (edubot->getBumper(0) ||
                       edubot->getBumper(1)) ? -SPEED/2 : SPEED/2;
        // para o edubot e move ele pra frente ou pra tras devagar
        edubot->stop();
        edubot->move(speed);
        edubot->sleepMilliseconds(1000);
        // espera um pouquinho enquanto move
        edubot->stop();
        // para de novo
        edubot->sleepMilliseconds(1000);
        // espera um segundo

        double angle = edubot->getTheta();
        // pega o angulo atual
        // rotaciona o edubot com base no angulo atual pra evitar colisao
        if (angle < 90) edubot->rotate(-angle);
        else if (angle < 180) edubot->rotate(-(angle - 90));
        else if (angle < 270) edubot->rotate(angle - 180);
        else edubot->rotate(angle - 270);

        edubot->sleepMilliseconds(2000);
```

```
        // espera dois segundos
        edubot->stop();
        // para depois de rotacionar

        return true;
        // retorna que houve colisao
    }
    return false;
    // retorna que nao houve colisao
}

int main() {
    // tenta conectar ao edubot
    if (edubot->connect()) {
        edubot->sleepMilliseconds(2000);
        // espera um pouquinho pra estabilizar a conexao
        // loop principal enquanto o edubot ta conectado
        while (edubot->isConnected()) {
            // se nao houver colisao
            if (!colisionRoutine()) {
                edubot->move(SPEED);
                // comeca a mover com a velocidade definida
                // continua movendo enquanto a distancia ta segura
                while (edubot->getSonar(3) >= MIN_DIST &&
                       (edubot->getSonar(6) <= MIN_DIST_SIDE ||
                        edubot->getSonar(0) <= MIN_DIST_SIDE));

                // se as distancias estiverem seguras, segue em frente
                if (edubot->getSonar(3) >= MIN_DIST &&
                    edubot->getSonar(6) >= MIN_DIST_SIDE &&
                    edubot->getSonar(0) >= MIN_DIST_SIDE) {
                    edubot->move(SPEED/2);
                    // vai mais devagar
                    edubot->sleepMilliseconds(TIME);
                    // move por um tempo definido
                    edubot->stop(); // para
                    edubot->sleepMilliseconds(1000);
                    // espera um segundo

                    // escolhe aleatoriamente entre girar
                    // 90 graus pra esquerda ou direita
                    switch (rand() % 3) {
                        case 0: edubot->rotate(90); break;
                        case 1: edubot->rotate(-90); break;
                    }
```

```
                    edubot->sleepMilliseconds(2000);
                    // espera dois segundos depois de girar
                    edubot->stop(); // para de novo
                    edubot->move(3 * SPEED); // continua movendo
                } else {
                    edubot->stop(); // para o robô
                    // se estiver muito perto de algo dos dois lados,
                    // gira 180 graus
                    if (edubot->getSonar(6) <= MIN_DIST_SIDE &&
                        edubot->getSonar(0) <= MIN_DIST_SIDE)
                        edubot->rotate(180);
                    else // senao, gira 90 graus pro lado mais seguro
                        edubot->rotate(edubot->getSonar(6) <= MIN_DIST_SIDE
                        ? -90 : 90);

                    edubot->sleepMilliseconds(2000);
                    // espera dois segundos depois de girar
                }
            }
        }
        edubot->disconnect()
        // desconecta o edubot quando terminar
    } else {
        std::cout << "Could not connect to robot!\n";
        // mensagem se a conexao falhar
    }

    return 0;
}
```

5 Conclusão

Trabalhar na programação do Edubot foi uma experiência muito interessante, cheia de desafios e aprendizados. No início, enfrentamos alguns problemas com o código, o que foi frustrante, mas ao mesmo tempo nos forçou a repensar a lógica e fazer ajustes importantes. Essa primeira tentativa falha acabou servindo para repensar no trabalho, mostrando que nem sempre as coisas saem como planejado, mas que isso faz parte do processo.

Depois de testar novamente, conseguimos fazer o robô funcionar como queríamos, o que foi uma grande satisfação para o grupo. Aprendemos que a prática de testar e ajustar é fundamental, especialmente em projetos que envolvem robótica e programação. Além disso, ficou claro como é importante ter paciência e persistência ao lidar com problemas técnicos.