UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIS GUILHERME MACHADO CAMARGO
PEDRO ALBERTO DE BORBA
RICARDO FARAH
STEFAN CAMPANA FUCHS
TELMO FRIESEN

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

ANÁLISE TECNOLÓGICA

CURITIBA

2013

LUIS GUILHERME MACHADO CAMARGO
PEDRO ALBERTO DE BORBA
RICARDO FARAH
STEFAN CAMPANA FUCHS
TELMO FRIESEN

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

Análise tecnológica apresentada à Unidade Curricular de Oficina de Integração 3 do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para aprovação.

CURITIBA

2013

SUMÁRIO

1 ANÁLISE TECNOLÓGICA

Nesta seção está explicitada, primeiramente, uma visão geral do projeto. Em seguida, há uma discussão detalhada a respeito dos requisitos de cada parte fundamental (estação base, sistema de comunicação, sistema embarcado). Por fim há uma enumeração das alternativas tecnológicas pesquisadas e das escolhidas para o preenchimento dos requisitos.

1.1 VISÃO GERAL DO PROJETO

O projeto, como foi idealizado, consiste em um robô controlado manualmente capaz de efetuar mapeamento 2D de ambientes. A estação base é um computador controlado e monitorado por um usuário humano. O utilizador será capaz de enviar comandos de movimentação ao robô (via teclado) e receber *feedback* do seu posicionamento e dos obstáculos detectados por ele. Além disso, as imagens em tempo real de uma câmera posicionada no robô – aspecto explicado mais à frente – poderão ser visualizadas pelo utilizador.

O sistema de comunicação deverá ter alcance máximo de 20 metros. Visto que toda a comunicação entre a estação base e o sistema embarcado será feita por um único canal, a velocidade de transmissão de dados deve ser suficiente para o envio de comandos de movimentação ao robô, recebimento de dados de leituras de sensores e recebimento de imagens da câmera em tempo real.

O sistema embarcado é constituido, em suma, pelo robô. Ele deve ser capaz de se mover para frente e para trás e girar para a esquerda e direita – em velocidades não muito altas, o que é suficiente, visto que velocidades elevadas dificultam o controle de movimentação pelo usuário. Uma visualização em tempo real do ambiente pelo usuário, tendo o objetivo de facilitar o controle de movimentação manual, poderá ser feita através de imagens geradas por uma câmera fixa instalada no robô.

O robô deve ser capaz de obter dados para cálculos (na estação base) da sua velocidade e deslocamento. Erros de medição em decorrência de escorregamento ou trepidação de rodas

devem ser atenuados, visando, dessa forma, a utilização futura do robô em condições não ideais de terreno. Obstáculos próximos – em uma distância mínima de 30 cm e máxima de 150 cm – devem ser detectados de modo a possibilitar a confecção do mapa 2D em tempo real na estação base.

1.2 REQUISITOS

1.2.1 Estação base

Esta seção descreve os requisitos da estação base, que foram elaborados de forma a satisfazer os objetivos do projeto.

- O *software* será executado em um computador pessoal.
  - O *software* deverá ser multiplataforma, ou seja, executar em diferentes sistemas operacionais (no mínimo Linux e Windows).
  - Preferencialmente bibliotecas e ferramentas livres (e gratuitas) deverão ser utilizadas no desenvolvimento do *software*.
- O *software* deve possuir uma interface gráfica.
  - Um utilizador, através da interface gráfica, será capaz de controlar o robô enviando comandos de movimentação especificados pelo teclado.
  - O usuário receberá a imagem em tempo real (preferencialmente com atrasos não muito consideráveis) de uma câmera fixa instalada no robô.
  - Os dados instantâneos de velocidade e posição do robô serão mostrados ao usuário na interface gráfica.
  - Um mapa 2D do caminho percorrido e dos obstáculos detectados pelo robô será gerado, na interface gráfica, à medida em que o robô se movimentar. O caminho percorrido por ele será representado por pontos que demonstrem visulamente a trilha percorrida por ele. Os obstáculos serão representados por pontos nos quais houve detecção de objetos pelos sensores. Todos os pontos representados no mapa serão gerados a partir de amostras em intervalos de tempo discretos de leituras de sensores do robô.
  - O mapa 2D gerado na interface poderá ser salvo em um arquivo, podendo ser posteriormente carregado.

1.2.2 Sistema de comunicação

Esta seção descreve os requisitos do sistema de comunicação entre a estação base e o sistema embarcado.

- Distância entre robô e estação base.
  - O sistema de comunicação deve possuir alcance máximo de 20 metros, de modo que ambientes de tamanho razoável possam ser mapeados.
- Velocidade e direção do fluxo de transmissão de dados.
  - A velocidade de transmissão do canal de comunicação deve ser suficiente para o envio de comandos de movimentação ao robô, recebimento de dados de leituras de sensores e recebimento de imagens da câmera em tempo real – visto que toda a comunicação entre a estação base e o sistema embarcado será feita por um único canal.
  - O fluxo de dados deve ser bidirecional (*full-duplex*).

1.2.3 Sistema embarcado

Esta seção descreve os requisitos do sistema embarcado (robô).

- Movimentação do robô.
  - O robô deve ser capaz de mover-se para frente, para trás e girar para a esquerda e direita em velocidades baixas.
- Controle de posicionamento e velocidade.
  - O robô deve ser capaz de obter dados que permitam calcular sua velocidade (linear e angular) e posição (deslocamento e rotação), enviando-os à estação base.
- Detecção de obstáculos.
  - O robô deverá ser capaz de detectar obstáculos próximos – com distância de no mínimo 30 cm e no máximo 150 cm – localizados ao seu redor, determinando a distância de cada objeto detectado.

1.3 ANÁLISE DE OPÇÕES TECNOLÓGICAS

Nesta seção está apresentada a análise das opções tecnológicas plausíveis para o atendimento dos requisitos. As alternativas pesquisadas e as escolhidas estão presentes.

1.3.1 Estação base

As alternativas pesquisadas para a estação base estão apresentadas a seguir.

1.3.1.1 Biblioteca para desenhos 2D

Tendo em vista o requisito de geração de um mapa 2D na interface gráfica da estação base, deve-se escolher uma biblioteca que permita realizar o desenho de formas geométricas e que possa ser integrada facilmente à interface gráfica. Deve também possuir meios simples de obter informações do mouse e teclado, para interatividade com o usuário.

Uma biblioteca importante disponível em Java que possui o recurso de produzir desenhos dinâmicos e integrá-los à interface gráfica é o Processing [REFERENCIA], *open-source*. Essa biblioteca foi a principal encontrada que seria capaz de satisfazer as necessidades de desenho do mapa 2D de forma simples. Por possuir inúmeras funções de desenho em alto nível, o trabalho de renderização dos gráficos seria consideravelmente simplificado. Além disso, há recursos que permitem o recebimento de informações de posicionamento de mouse e comandos do teclado. Por ser constituído basicamente de um *Applet* Java, o Processing pode facilmente ser integrado a componentes do *swing*.

Outra biblioteca para a confecção de desenhos em 2D é o Cairo [REFERENCIA], que é *open-source*. Essa biblioteca possui recursos em alto nível para renderização de formas, assim como o Processing. Porém, a integração com a interface gráfica é dependente na biblioteca externa utilizada para tal.

Na Tabela 1 está presente uma comparção entre as duas bibliotecas. Ambas as bibliotecas são dependentes de linguagens específicas, como demonstrado na tabela.

A escolha da biblioteca de desenhos foi feita em conjunto com a escolha de linguagem de programação. A biblioteca escolhida, dentre essas opções foi o Processing, visto que pode ser facilmente integrada à interface do Java.

Tabela 1: Comparação entre Bibliotecas para desenhos 2D.

Característica	**Cairo**	**Processing**
Linguagem de programação	C e C++	Java
Integração com interface gráfica	Sim (depende da biblioteca de GUI utilizada)	Sim (no *swing* do Java)
Ferramentas de interação com o usuário	Sim	Sim

1.3.1.2 Linguagem de programação

Nessa etapa de avaliação das opções, a escolha de uma boa linguagem de programação que atenda aos requisitos é fundamental. Abaixo está presente uma lista dos aspectos desejáveis da linguagem:

- Deve ser multiplataforma (ao menos compatível com Linux e Windows sem muitas modificações);
- Deve possuir orientação a objetos;
- Deve possuir recursos multiplataforma e *open-source* para o desenvolvimento de interface gráfica;
- Deve ter a disponibilidade de ferramentas *open-source* e multiplataforma para a criação visual da interface gráfica, dessa forma agilizando o processo de desenvolvimento;
- Deve possuir recursos, integrados ou em bibliotecas *open-source*, para o desenvolvimento de desenhos dinâmicos (para a geração do mapa 2D). Os desenhos devem ser facilmente integráveis à interface gráfica.

Abaixo está presente uma comparação entre duas linguagens, o C++ e Java, que são amplamente usadas atualmente.

Java

O Java [REFERENCIA] é uma linguagem concebida de início como sendo orientada a objetos. A maneira com que é feita a compilação e execução do código permite que muito facilmente programas sejam rodados em diferentes plataformas (Linux, Windows, Mac, entre outros). O processo de compilação do código gera os chamados *bytecodes*, que são instruções a serem interpretadas pela *Java Virtual Machine* (JVM). A grande vantagem é que o JVM possui disponibilidade multiplataforma, possuindo ainda frequente manutenção pelos desenvolvedores.

Há disponibilidade, na API do Java, da biblioteca *swing* – ferramenta completa para a criação de interfaces gráficas (GUI) interativas. Existem ferramentas visuais *open-source* que consideravelmente agilizam o processo de desenvolvimento de interfaces *swing*, entre elas o *NetBeans* e o *Eclipse* (através de plugins ou extensões).

Para efetuar desenhos em 2D e integrá-los à interface gráfica, a biblioteca do Processing (explicada anteriormente) está disponível nessa linguagem.

C++

O C++ é uma linguagem orientada a objetos, que foi evoluída a partir da linguagem C. A compilação de código no C++ deve ser feita especificamente para cada plataforma em que o programa será utilizado. De uma perspectiva prática, certas seções de código frequentemente necessitam de adaptações manuais para cada plataforma e sistema operacional, o que gera retrabalho e gastos de tempo adicionais.

Recursos para desenvolvimento visual de interfaces gráficas são disponíveis através de bibliotecas e ferramentas externas. Como o C++ não possui recursos de interface gráfica na própria API, essa é uma dificuldade que se faz presente no quesito da portabilidade entre diferentes sistemas.

Para a integração de desenhos 2D à interface gráfica, a biblioteca Cairo (explicada anteriormente) pode ser utilizada com essa linguagem. Porém, a integração é dependente da biblioteca de GUI utilizada.

Escolha da equipe: O Java foi a linguagem escolhida para o desenvolvimento do *software* da estação base, uma vez que preenche satisfatoriamente os requisitos do projeto. Notavelmente, há a facilidade em portar, sem adaptações, programas para diferentes plataformas – ao contrário do que ocorre com o C++.

1.3.2 Sistema de comunicação

Tabela 2: Comparação entre linguagens de programação.

Característica	**C++**	**Java**
Multiplataforma (Linux e Windows)	Sim (com adaptação)	Sim (sem adaptação)
Orientação a objetos	Sim	Sim
Recursos multi-plataforma e *open-source* para desenvolvimento de interface gráfica (GUI)	Sim (com bibliotecas externas)	Sim (integrado à API da linguagem)
Ferramentas *open-source* e multiplataforma para criação visual de interface gráfica	Sim (ferramentas externas)	Sim (ferramentas externas)
Recursos *open-source* para desenvolvimento de desenhos dinâmicos, facilmente integráveis à interface gráfica	Sim (biblitoeca externa, integração à interface gráfica dependente da GUI utilizada)	Sim (biblioteca externa)

REFERÊNCIAS