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 e 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 de um ponto de vista geral, consiste em um robô controlado manualmente que seja capaz de efetuar mapeamento em duas dimensões de ambientes. Um usuário humano controlará e monitorará um computador – a estação base – através do qual poderão ser enviados comandos de movimentação ao robô (via teclado). Informações sobre o posicionamento do robô e dos obstáculos detectados por ele serão recebidas na estação base em tempo real. Imagens instantâneas de uma câmera posicionada no robô – aspecto explicado mais à frente – poderão ser visualizadas pelo utilizador.

O sistema de comunicação deverá ter, ao menos, 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 e o tipo de fluxo de transmissão de dados devem ser suficientes 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. Uma visualização em tempo real do ambiente pelo utilizador, tendo o objetivo de facilitar o controle de movimentação manual, poderá ser feita através de imagens instantâneas geradas por uma câmera fixa instalada no robô.

O robô deve ser capaz de obter dados para cálculos (na estação base) de sua velocidade e deslocamento. Um aspecto a ser trabalhado em relação a isso é a atenuação de erros em decorrência de escorregamento, giros em falso ou trepidação de rodas, 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 programa deverá ser executado razoavelmente em computadores com recursos de *hardware* comparáveis aos padrões atuais (pelo menos 2GB de memória RAM DDR2 ou melhor e processador *dual-core*).
  - 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 houver movimentação do mesmo. O caminho percorrido pelo robô será representado visualmente por pontos, gradualmente posicionados no mapa. Os obstáculos serão representados por marcações nas localidades onde houve detecção de objetos por sensores. Todos os pontos representados no mapa serão gerados a partir de amostras obtidas em intervalos de tempo discretos.

  - 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, ao menos, alcance máximo de 20 metros sem fios – de modo que ambientes de tamanho razoável possam ser mapeados.
- Velocidade e direção do fluxo de transmissão de dados.
  - Toda a comunicação entre a estação base e o sistema embarcado será feita por um único canal e, portanto:
  - 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;
  - O fluxo de dados deve ser bidirecional (*full-duplex*).
- Protocolo de transporte.
  - A tecnologia utilizada para a comunicação deve permitir fácil utilização do protocolo de transporte TCP. Como as leituras de sensores devem obrigatoriamente ser recebidas na estação base na mesma ordem em que forem enviadas pelo robô (e também os comandos de movimentação enviados pela estação base devem chegar ao robô em ordem), o uso desse protocolo de transporte simplificará muito a implementação do protocolo em alto nível entre os dois pontos. Outro aspecto positivo do TCP é que além de garantir a ordem de chegada, existem mecanismos de detecção de perdas de pacotes – que efetuam o reenvio caso necessário.

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.

- Controle de posicionamento e velocidade.

  - O robô deve ser capaz de obter dados que permitam calcular sua velocidade (linear e angular) e posição atual (deslocamento e rotação em relação à posição inicial). Deve ser capaz de enviar os dados 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 para cada parte do projeto estão explicitadas a seguir.

1.3.1 Estação base

As alternativas pesquisadas para a estação base estão apresentadas nesta subseção.

1.3.1.1 Biblioteca para desenhos 2D

Tendo em vista que um dos requisitos é a geração de um mapa em duas dimensões na estação base, deve-se escolher uma biblioteca que permita realizar o desenho de formas geométricas diversas e que possa ser integrada facilmente à interface gráfica. Ela deve também possuir meios simples de obter informações do mouse e teclado, para interatividade com o usuário.

Uma biblioteca interessante disponível em Java que possui o recurso de produzir desenhos dinâmicos (e integrá-los a interfaces gráficas) é o Processing (PROCESSING, 2013), *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, na biblioteca existem recursos que permitem o recebimento de informações de posicionamento do mouse e de comandos do teclado. Por ser constituído basicamente de um *Applet* Java, o Processing pode facilmente ser integrado a componentes do Swing – biblioteca de interface gráfica (GUI) do Java.

Outra biblioteca para a confecção de desenhos em 2D é o Cairo (CAIRO, 2013), que é *open-source*, disponível nas linguagens C e C++. Ele possui recursos em alto nível para renderização de formas e interação com o usuário, assim como o Processing. Nos aspectos gerais as duas ferramentas são muito semelhantes. A integração do Cairo com a interface gráfica, porém, é dependente na biblioteca externa de GUI utilizada para tal.

Um aspecto importante a notar é que ambas as bibliotecas foram desenvolvidas e otimizadas para terem bom desempenho em máquinas atuais – o que é desejável tendo em vista os requisitos. Na Tabela 1 está presente uma comparação entre as duas bibliotecas.

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 (na biblioteca Swing do Java)
Ferramentas de interação com o usuário	Sim	Sim

A escolha da biblioteca de desenhos foi feita em conjunto com a escolha de linguagem de programação. A biblioteca escolhida, dentre as duas opções, foi o Processing, visto que a integração a interfaces gráficas do Java é muito simples.

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 interfaces gráficas;

- 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 descrição das duas linguagens, o C++ e Java, atualmente utilizadas em inúmeras aplicações, e que são potenciais alternativas ao projeto. A Tabela 2 sumariza os recursos de cada uma.

Java

O Java (JAVA, 2013) 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, e a manutenção pelos desenvolvedores é frequente.

Há disponibilidade, na API do Java, da biblioteca Swing – que contém recursos completos para a criação de interfaces gráficas (GUI) interativas. Existem ferramentas visuais de código aberto que consideravelmente agilizam o processo de desenvolvimento de interfaces Swing, entre elas o NetBeans (NETBEANS, 2013) e o Eclipse (ECLIPSE, 2013), através de plugins ou extensões.

Para o preenchimento do requisito de confecção de desenhos em 2D com integração à interface gráfica, a biblioteca do Processing (explicada anteriormente na Subseção 1.3.1.1) está disponível nessa linguagem.

C++

O C++ é uma linguagem orientada a objetos, que foi desenvolvida a partir da linguagem C. A compilação de código no C++ deve ser feita especificamente para cada plataforma em que os programas desenvolvidos forem utilizados. 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 estão disponíveis através de bibliotecas e ferramentas externas. O C++ não possui recursos de interface gráfica na própria API. Deve-se notar que esse é um aspecto que adiciona complexidade ao portar programas entre

diferentes sistemas.

Para a confecção de desenhos 2D e incorporação dos mesmos à interface gráfica, a biblioteca Cairo (explicada anteriormente na Subseção 1.3.1.1) pode ser utilizada com essa linguagem. A possibilidade de haver integração com a interface, porém, 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. A escolha do Java foi feita em conjunto com a escolha da biblioteca do Processing. Notavelmente, há a facilidade em portar, sem adaptações, programas para diferentes plataformas, processo este que é mais complexo no C++. Com relação ao quesito de desempenho em computadores atuais, a linguagem escolhida é satisfatória, visto que há manutenção constante da implementação das bibliotecas e da máquina virtual do Java pelos desenvolvedores – que buscam, entre outros aspectos, otimizar a linguagem para tecnologias atuais.

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)

1.3.2 Sistema de comunicação

Na Tabela 3 está presente uma comparação entre diferentes tecnologias de comunicação sem fios. O Wi-Fi é o recurso mais atrativo em todos os aspectos que foram comparados, preenchendo satisfatoriamente os requisitos do sistema de comunicação. Sua velocidade e alcance são suficientes para satisfazer as necessidades, e o fluxo de dados pode ser *full-duplex*. Notavelmente, o Wi-Fi é o único sistema comparado que oferece a possibilidade (com simplicidade)

de uso do protocolo TCP – o que é um requisito importante para o desenvolvimento ágil e satisfatório do projeto.

Tabela 3: Comparação entre tecnologias de comunicação sem fios.

Característica	**802.11g (Wi-Fi)**	**Rádio Frequência (RF)**	**Bluetooth**
Distância máxima de alcance	50-100 metros	30-100 metros	10 metros
Velocidade de transmissão máxima	54 Mbits/s	2 Mbits/s	1 Mbits/s
Fluxo de dados *full-duplex*	Sim	Sim	Sim
Possibilidade e simplicidade de uso de TCP	Sim	Não	Não

1.3.3 Sistema embarcado

Nesta seção apresentamos as alternativas pesquisadas para o sistema embarcado, levando em conta os requisitos já apresentados anteriormente.

1.3.3.1 Movimentação do robô

O sistema de movimentação do robô, incluindo motores, acionadores, drivers de potência e rodas não foram pesquisados pois já estão implementados no robô e atendem aos requisitos especificados nas seções anteriores. Sendo assim utilizaremos um chassi de 40 cm de largura por 50 cm de comprimento, duas rodas de tração e uma roda guia. As rodas de tração estão dispostas na parte posterior do robô, possuindo 20 cm de diâmetro e 4 cm de largura. O chassi está equipado com 2 motores Bosch FPG 12V, 2 baterias Unybatt 12V-7,2 Ampére-hora e duas pontes H L298 (MARIN et al., 2012). A disposição dos itens no robô pode ser vista na figura 1.

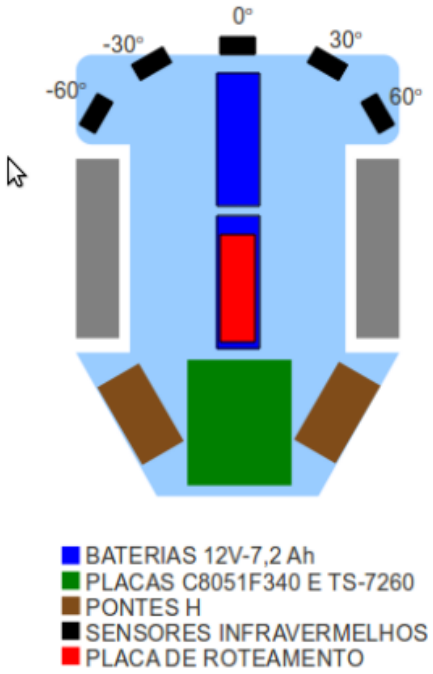

Figura 1: Disposição dos itens no robô

Fonte: (MARIN et al., 2012)

1.3.3.2 Odometria

Para a obtenção da direção e sentido do movimento do robô, assim como aceleração, velocidade e posição, pode-se optar por diversos tipos de tecnologias ou pela associação delas. Abaixo descrevemos as principais delas.

- **Encoder:** Ligado ao eixo da roda do robô, conta a quantidade de voltas dadas pela roda. Permite assim calcular a distancia percorrida pelo robô. Caso sejam conectados dois encoders, um em cada roda, podemos obter também a direção do movimento pela diferença entre a contagem de voltas de cada roda.
- **GPS:** Utiliza sinais de satélites para obter as coordenadas geográficas do robô. A direção e o sentido podem ser obtidos pela comparação das leituras com as anteriores.
- **Acelerômetro:** pode utilizar a tecnologia chamada **MEMS** para medir a aceleração sofrida pelo componente.
- **Giroscópio:** pode utilizar a tecnologia chamada **MEMS** para medir a aceleração angular sofrida pelo componente.

- **Bussola:** Utiliza os campos magnéticos da terra para obter a direção com relação aos polos magnéticos da terra.

A seguir, na tabela 4, comparamos as tecnologias apresentadas anteriormente.

Tabela 4: Comparação entre tecnologias para odometria.

Característica	**Encoder**	**GPS**	**Acelerômetro**	**Giroscópio**	**Bussola**
Sujeito a influencias externas	Deslizamentos	Não	Não	Não	Campo magnético dos motores
Ambiente de operação	Interno / Externo	Externo	Interno / Externo	Interno / Externo	Interno / Externo
Posicionamento	Relativo	Absoluto	Relativo	Relativo	Relativo

Com base no comparativo da tabela 4 optamos por utilizar um acelerômetro em conjunto com um giroscópio para obter os dados para odometria. Encoders estão sujeitos a erros causados por deslizamento nas rodas, GPS apenas funciona em ambientes externos e a bussola pode ser influenciada pelo campo magnético dos motores. Como o robô já apresenta encoders instalado, utilizaremos também os dois encoders, possibilitando assim aumentar a precisão e confiabilidade dos dados de odometrias. Justificada nossa escolha por acelerômetros e giroscópios, na tabela 5 fazemos um comparativo entre as opções de menor custo disponíveis no mercado. Listamos na tabela apenas as alternativas que possuem placas de desenvolvimento pois acelerômetros e giroscópios são geralmente são vendidos em encapsulamento LGA, os quais são de difícil soldagem.

MPU-6050

Com base na tabela 5 escolhemos o modelo MPU-6050 da IvenSense principalmente devido ao seu baixo custo: \$8.78. Este modelo possui um acelerômetro de 3 eixos, um giroscópio e 3 eixos e entradas para uma bussola externa integrados em um único chip (EVENSENSE, 2013). A faixa de operação para o acelerômetro é de $\pm 2g ou \pm 4g$ e para o giroscópio é de $\pm 250^o/seg$ ou $\pm 500^o/seg$. A sensibilidade do acelerômetro é de $16384 LSB/g$ e $8192 LSB/g$. A sensibilidade do giroscópio é de $131 LSB/^o/seg$ e $65.5 LSB/^o/seg$. A interface de comunicação do módulo suporta o protocolo I2C. O módulo contendo o chip MPU-6050 e alguns componentes necessários para seu funcionamento pode ser visto na figura 2.

Tabela 5: Comparação entre acelerômetros/giroscópios para odometria.

Modelo	**Fabricante**	**Acel.**	**Giro.**	**Faixa**	**Interface**	**Preço**
STEVAL-MKI009V1	STMicroeletronics	3x	-	$\pm 2g / \pm 6g$	I2c / SPI	$23.94
ATAVRSBIN1	Atmel	1x	-	-	I2c	$26.25
KIT3803 MMA7660FC	Freescale	3x	-	$\pm 1.5g$	I2C	$35.0
ATAVRSBIN1	Atmel	-	1x		I2C	$26.25
MPU-6050	IvenSense	3x	3x	$\pm 2g / \pm 4g$ $\pm 250^o/seg /$ $\pm 500^o/seg$	I2C	$8.78
MKI086V1	STMicroeletronics	1x	$\pm 30^o/seg$	Analog		$31.50
STEVAL-MKI094V1	STMicroeletronics	-	3x	$\pm 400^o/seg$	Analog	$31.50
ATAVRSBIN1	Atmel	1x	1x		I2C	$26.25
DM240316	Zena	3x	3x		RF	$99.99

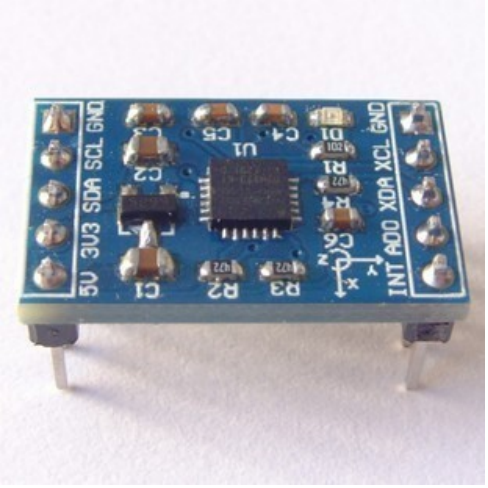

Figura 2: Placa de desenvolvimento contendo o chip MPU-6050

Fonte: (EVENSENSE, 2013)

Encoder Optico HEDS-9700

Como já foi dito, utilizaremos também os encoders já existentes no robô. O robô está equipado com dois encoders HEDS-9700. Esses encoders geram em sua saída uma onda quadrada à medida em que o encoder é rotacionado, sendo 1800 pulsos gerados em uma rotação (AGILENTTECHNOLOGIES, 2002). A forma de onda da saída do encoder pode ser vista na figura 3. Podemos ver na figura que o encoder possui duas saidas, A e B com defasamento ϕ entre

elas. Com as duas saídas podemos determinar o sentido de rotação. Porém na implementação existente esse recurso não é utilizado.

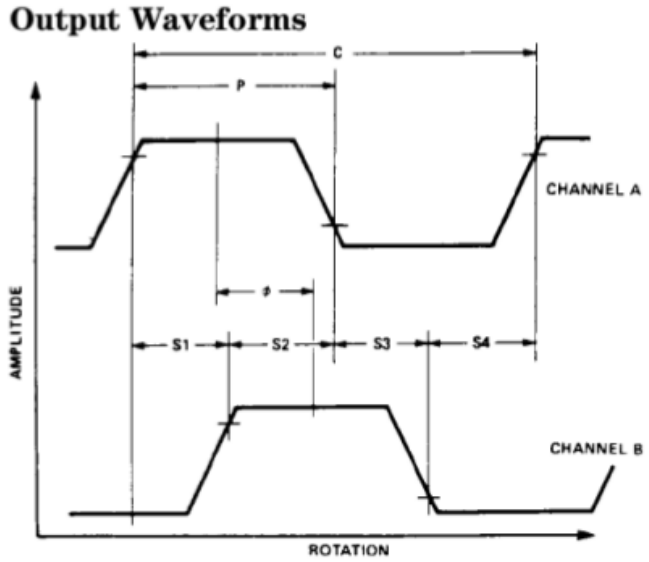

Figura 3: Forma de onda de saída do encoder
Fonte: (AGILENTTECHNOLOGIES, 2002)

1.3.3.3 Detecção de obstáculos

Sensor de proximidade Infra Vermelho IR 2Y0A02F98

A detecção de obstáculos, que é um requisito deste robô, será atendido pelos sensores de Infra vermelho já existentes no robô. Logo serão utilizados sensores IR 2Y0A02F98 (SHARPCORPORATION, 2006). Este modelo é pouco influenciado pelas cores dos objetos detectados devido ao método de medição baseado em triangulação. Na figura 4 podemos ver isso. A linha tracejada é a resposta para reflexão em um papel cinza e a linha contínua é a resposta para reflexão em um papel branco.

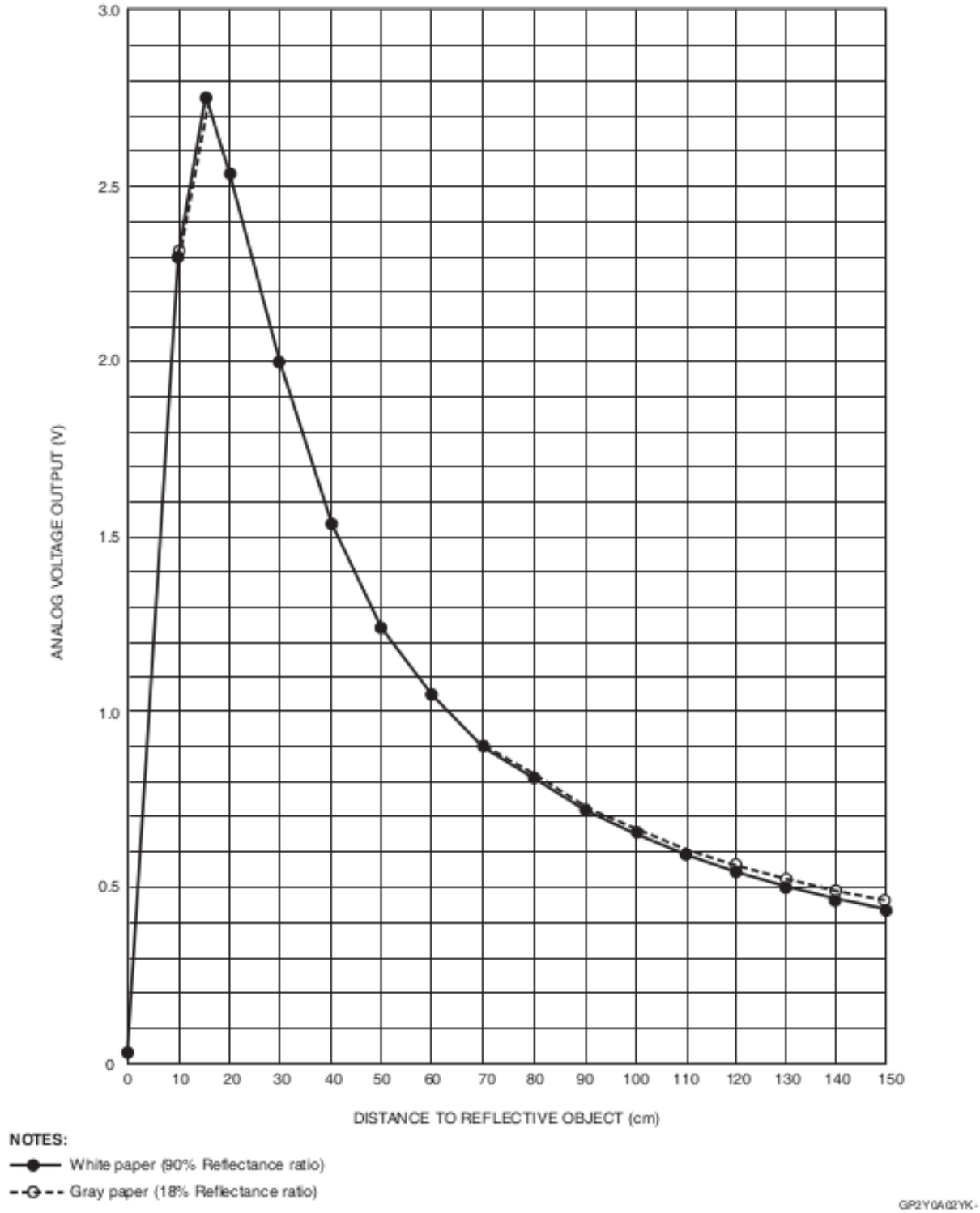

Figura 4: Curva de resposta do encoder

Fonte: (SHARPCORPORATION, 2006)

1.3.3.4 Microcontrolador

A interface entre os sensores, atuadores e a placa TS-7260 já existente no robô (MARIN et al., 2012) será feita por um sistema micro controlado. Uma vez escolhidos os sensores, o sistema micro controlado deve possuir as interfaces adequadas para comunicação com os sensores e também para comunicação com o hardware já existente no robô, como o sistema de acionamento dos motores e interface com a placa TS-7260. Desta forma, na tabela 6 listamos os requisitos para escolha do microcontrolador. Na tabela 7 listamos os requisitos que não são obrigatórios, porém desejáveis para o microcontrolador.

Tabela 6: Requisitos para escolha do microcontrolador.

Requisito	**Justificativa**
Interface I2C	Comunicação com acelerômetro e giroscópio
Geração de PWM em 4 canais	Acionamento dos motores pelas pontes H
Interface Serial	Comunicação com a placa TS-7260
Interrupções em 2 canais	Leitura do valor dos encoders
Conversor AD em 5 canais	Leitura dos sensores de IR

Tabela 7: Requisitos desejáveis para escolha do microcontrolador.

Requisito desejável	**Justificativa**
Desenvolvimento em plataforma livre	Diminui o custo de softwares para desenvolvimento
32 bits	Facilita manipulações numéricas na programação, diminuindo esforço e custo para programação
2 Interfaces seriais ou JTAG	Utilização para debug ou logs
Solução integrada	Redução do tamanho da placa e quantidade de componentes, diminuindo assim o custo e melhorando a organização e disposição dos componentes

Na tabela 8 listamos diversas opções de microcontroladores que foram pesquisados para o projeto. Todas as opções atendem aos requisitos da tabela 6, e são 32 bits.

Dentre as opções listadas na tabela 8 escolhemos a opção **LPC2103**. Esta escolha foi feita baseada principalmente no custo do microcontrolador. Porém ao compará-lo com o **SIM3C146** vemos que esta última opção possui desempenho melhor com custo menor. Nossa escolha pelo **LPC2103** e não pelo **SIM3C146** justifica-se pela melhor documentação e dispo-

nibilidade de recursos para o **LPC2103**. A documentação fornecida pelo fabricante do microcontrolador **LPC2103** é mais completa, e por já estar a mais tempo no mercado a quantidade de informações e recursos disponíveis na internet é maior.

LCP2103

O **LCP2103** é um microcontrolador baseado na arquitetura ARM7TDMI-S da NXP (NXP, 2013). Este microcontrolador pode operar em até $70MHz$ executando a $63MIPS$. O Microcontrolador possui 2 interfaces I2C, 2 interfaces seriais, até 14 saídas de PWM, até 13 canais de interrupções externas, 8 canais de conversão para um conversor analógico digital de 10 bits, 32kbytes de memória FLASH para código e 8kbytes de memória RAM. Ele também suporta *debug* via JTag por meio de um *debugger* JTag externo. O custo desse microcontrolador é de $6.16 (CORPORATION, 2013).

O microcontrolador escolhido também pode ser programado utilizando o protocolo ISP por meio de ferramentas livres como o lpc21isp (LPC21ISP, 2013). Para geração do código hexadecimal utilizado pelo lpc21isp basta compilar o código em C utilizando o GCC (GCC, 2013). Portanto o **LCP2103** além de atender aos requisitos propostos anteriormente também atende aos requisitos desejáveis que foram propostos na tabela 7.

Tabela 8: Comparativo entre microcontroladores.

Fonte: Dados obtidos de (CORPORATION, 2013)

uC	**STM32F103C6T7A**	**PIC32MX320F128H**	**LPC2103**
Fabricante	STMicroelectronics	Microchip Technology	NXP Semiconductors
Arquitetura	ARM® Cortex™-M3	MIPS32® M4K™	ARM7
Core	32bits	32-Bit	16/32-Bit
Velocidade	72MHz	80MHz	70MHz
MIPS	90	124.8	63
I2C	1	2	2
PWM	12	5	14
UART	2	2	2
Einterrupt	16	5	13
FLASH	32k	128k	32k
RAM	10k	16k	8k
Adc	10x12b	16x10b	8x10b
JTAG	sim	sim	sim
Custo	$6.27	$6.26	$6.16
uC	**MCF52210CAE66**	**AT32UC3C264C**	**SIM3C146**
Fabricante	Freescale Semiconductor	Atmel	Silicon Laboratories Inc
Arquitetura	Coldfire V2	AVR	ARM® Cortex™-M3
Core	32-Bit	32-Bit	32-Bit
Velocidade	66MHz	66MHz	80MHz
MIPS	75.9	98.34	100
I2C	2	3	2
PWM	4	8	8
UART	3	1	2
Einterrupt	7	7	16
FLASH	64k	64k	64k
RAM	16k	16k	16k
Adc	8x12b	11x12b	28x12b
JTAG	sim	sim	SIM3C146-B-GQ
Custo	$7.1	$9.14	$6.1

REFERÊNCIAS

AGILENTTECHNOLOGIES. **Small Optical Encoder Modules HEDS-9700 Series**. 2002. Disponível em: <http://www.digchip.com/datasheets/parts/datasheet/021/QEDS-9871-pdf.php>. Acesso em: 28 de Fevereiro de 2013.

CAIRO. 2013. Disponível em: <http://www.cairographics.org/>.

CORPORATION, D. **DigiKey Corporation**. 2013. Disponível em: <http://www.digikey.com>. Acesso em: 18 de Fevereiro de 2013.

ECLIPSE. 2013. Disponível em: <http://eclipse.org/>.

EVENSENSE. 2013. Disponível em: <http://www.invensense.com/mems/gyro/mpu6050.html>.

GCC. 2013. Disponível em: <http://gcc.gnu.org/>. Acesso em: 18 de Fevereiro de 2013.

JAVA. 2013. Disponível em: <www.java.com>.

LPC21ISP. 2013. Disponível em: <http://sourceforge.net/projects/lpc21isp/>. Acesso em: 18 de Fevereiro de 2013.

MARIN, A. J.; BORGES, J. C. N.; WERGRZN, Y. A. **Desenvolvimento de robô móvel e análise qualitativa de algoritmos de navegação fuzzy**. Curitiba, 2012.

NETBEANS. 2013. Disponível em: <http://netbeans.org/>.

NXP. **LPC2103 Overview**. 2013. Disponível em: <http://www.nxp.com/products/microcontrollers/arm7/LPC2103FBD48.html>. Acesso em: 28 de Fevereiro de 2013.

PROCESSING. 2013. Disponível em: <www.processing.org>.

SHARPCORPORATION. **GP2Y0A02F98YK Datasheet**. 2006. Disponível em: <http://www.mindsensors.com/index.php?module=documents&JAS_DocumentManager_op=downloadFile&JAS_File_id=335>. Acesso em: 1 de Agosto de 2011.