

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

ALEXANDRE JACQUES MARIN
BRUNO WEINGRABER
LUCAS CAMPIOLO PAIVA
YURI ANTIN WERGRZN

DESENVOLVIMENTO DO PROJETO BELLATOR

RELATÓRIO DE CONCLUSÃO

CURITIBA
2010

ALEXANDRE JACQUES MARIN

BRUNO WEINGRABER

LUCAS CAMPIOLO PAIVA

YURI ANTIN WERGRZN

DESENVOLVIMENTO DO PROJETO BELLATOR

Relatório apresentado à Disciplina de Oficina de Integração 3, do Departamento Acadêmico de Informática - DAINF - e do Departamento Acadêmico de Eletrônica - DAELN - do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para finalização da disciplina.

CURITIBA

2010

Sumário

1	Introdução	6
1.1	Resumo	6
1.2	Abstract	6
1.3	Objetivos	6
1.4	Motivação e Justificativa	7
2	Fundamentação Teórica	8
2.1	Visão geral do sistema	8
2.2	Camada Supervisória	10
2.2.1	Hardware	10
2.2.2	Software	11
2.3	Camada de Alto Nível	11
2.3.1	Hardware	11
2.3.2	Software	13
2.4	Camada de Baixo Nível	13
2.4.1	Hardware	13
2.4.2	Software	14
2.5	Protocolo de Comunicação	15
2.6	Bibliotecas Externas	16
3	Projeto	17
3.1	Projeto do Software Supervisor Remoto	17
3.1.1	Levantamento de Requisitos	17
3.1.2	Estudo de Casos de Uso	18
3.1.3	Diagrama de Classes	19
3.2	Projeto do Hardware	20
3.2.1	C8051F340DK	20
3.2.2	Sensor IR 2Y0A02F98	22
3.2.3	Placa de Roteamento	26
3.2.4	Webcam	28
3.2.5	Robô Bellator	28
3.2.6	PC embarcado	29
3.2.7	Interconexão do Hardware	30
3.2.8	Joystick	33
4	Execução	35
4.1	Estado do Sistema	35
4.2	Software Supervisor Remoto	35
4.2.1	Util	36

4.2.2	Principal	36
4.2.3	Controlador	38
4.2.4	Comunicação	39
4.2.5	Interface Gráfica	40
4.3	Software de Baixo Nível	41
4.3.1	Migração para C8051F340DK	41
4.3.2	Funcionamento do Software	42
5	Considerações Finais	45
5.1	Resultados	45
5.2	Conclusão	45
5.3	Trabalhos Futuros	46
Referências		47
Anexo A - Protocolo		49

Lista de Figuras

1	Visão Geral do Sistema	9
2	Diagrama em Blocos da Camada Supervisória	11
3	Diagrama em Blocos da Camada de Alto Nível	12
4	Diagrama em Blocos da Camada de Baixo Nível	14
5	Diagrama de Casos de Uso.	19
6	Diagrama de Classes.	20
7	Diagrama em blocos do kit C8051F340DK.	22
8	Curva de resposta do sensor de distância.	23
9	Diagrama esquemático do regulador de tensão dos sensores de distância.	24
10	Dimensões do sensor GP2Y0A02F98YK.	25
11	Diagrama da Placa de Roteamento	27
12	Foto do robô Bellator	29
13	VIA EPIA M-Series MiniITX Mainboard	30
14	Posicionamento dos Elementos do Robô - Parte 1	31
15	Posicionamento dos Elementos do Robô - Parte 2	32
16	Joystick de Sony Playstation 2.	34
17	Interface do software Supervisor Remoto.	36
18	Funcionamento do módulo Controle Principal.	37
19	Funcionamento do módulo Comunicação.	40
20	Interface do software Supervisor Remoto em funcionamento. .	41

1 Introdução

1.1 Resumo

Este trabalho descreve o planejamento e a realização do projeto Bellator, que visou implementar mais algumas funcionalidades para a já existente plataforma robótica Bellator. O foco do projeto realizado por esta equipe foi, partindo do trabalho já realizado na plataforma anteriormente, a implementação de um software supervisor remoto em linguagem Java que permitisse que o robô fosse controlado remotamente via wireless através de um joystick ligado a um PC. Além disso, o software disponibiliza para a navegação a imagem da webcam e os dados de sensores de distância instalados no robô. O projeto também incluiu uma adaptação do controle de mais baixo nível do robô, realizado por uma placa com processador 8051, para que fossem adicionados mais sensores de distância, proporcionando um melhor controle da situação do robô para o usuário do supervisor remoto.

1.2 Abstract

This paper describes the planning and execution of the Bellator project, which aimed to implement some more features to the existing robotic platform Bellator. The focus of the project undertaken by this team was, starting with the work previously done on the robotic platform, the software implementation of a remote supervisor in the Java programming language, which allowed the robot to be remotely controlled by a joystick connected to a PC via wireless. In addition, the software provides navigation aid through the webcam image and the data from the distance sensors installed on the robot. The project also included an adaptation of the lower level control of the robot, done by 8051 processor, to add more distance sensors, providing better navigation of the robot to the user of the remote supervisor.

1.3 Objetivos

Contribuir para o aprimoramento da plataforma robótica Bellator, para tal realizar as seguintes tarefas:

- Migrar o controle da camada de baixo nível do robô da placa atual para a C8051F340DK.
- Adicionar ao robô 6 novos sensores ópticos.
- Desenvolver um software para a camada supervisória do robô que permita que este seja controlado remotamente.

- Documentar todo o trabalho realizado adequadamente para que possa ter continuidade no futuro.

1.4 Motivação e Justificativa

A maior motivação para a realização deste trabalho é, tendo em vista o contexto de sua realização, o aprendizado. No caso específico da disciplina para a qual o projeto está sendo realizado, este aprendizado refere-se não apenas aos conteúdos adquiridos na implementação dos produtos propostos mas, também, aprendizado sobre planejamento e gerenciamento de projeto. A equipe tem também o desafio de realizar um projeto partindo do trabalho realizado por várias outras pessoas, e que na maior parte das vezes não foi muito bem documentado. Conhecendo essas dificuldades, é dever da equipe finalizar o projeto deixando uma documentação muito melhor do que a encontrada, visto que a plataforma Bellator tem potencial para abrigar muitos outros projetos no futuro.

Assim, a justificativa para a realização deste projeto torna-se simples: além de ser uma ótima oportunidade de aprendizado para a equipe, tanto nos termos técnicos da plataforma quanto no gerenciamento de projetos, este trabalho tem como objetivo continuar o desenvolvimento da plataforma robótica, permitindo a realização de trabalhos cada vez mais elaborados.

2 Fundamentação Teórica

2.1 Visão geral do sistema

O sistema consiste de três camadas lógicas: Baixo Nível, Alto Nível e Supervisória. A camada de baixo nível é responsável pelo controle e realimentação do hardware instalado na plataforma robótica Bellator. Já a camada de alto nível é responsável por proporcionar um meio de comunicação entre a camada supervisória e a de Baixo Nível, além de gerar realimentação visual, através de stream de vídeo, para a camada supervisória. Finalmente, a camada supervisória deve providenciar ao usuário uma interface visual simples para o controle da plataforma robótica. A figura a seguir mostra esta configuração na forma de um diagrama em blocos.

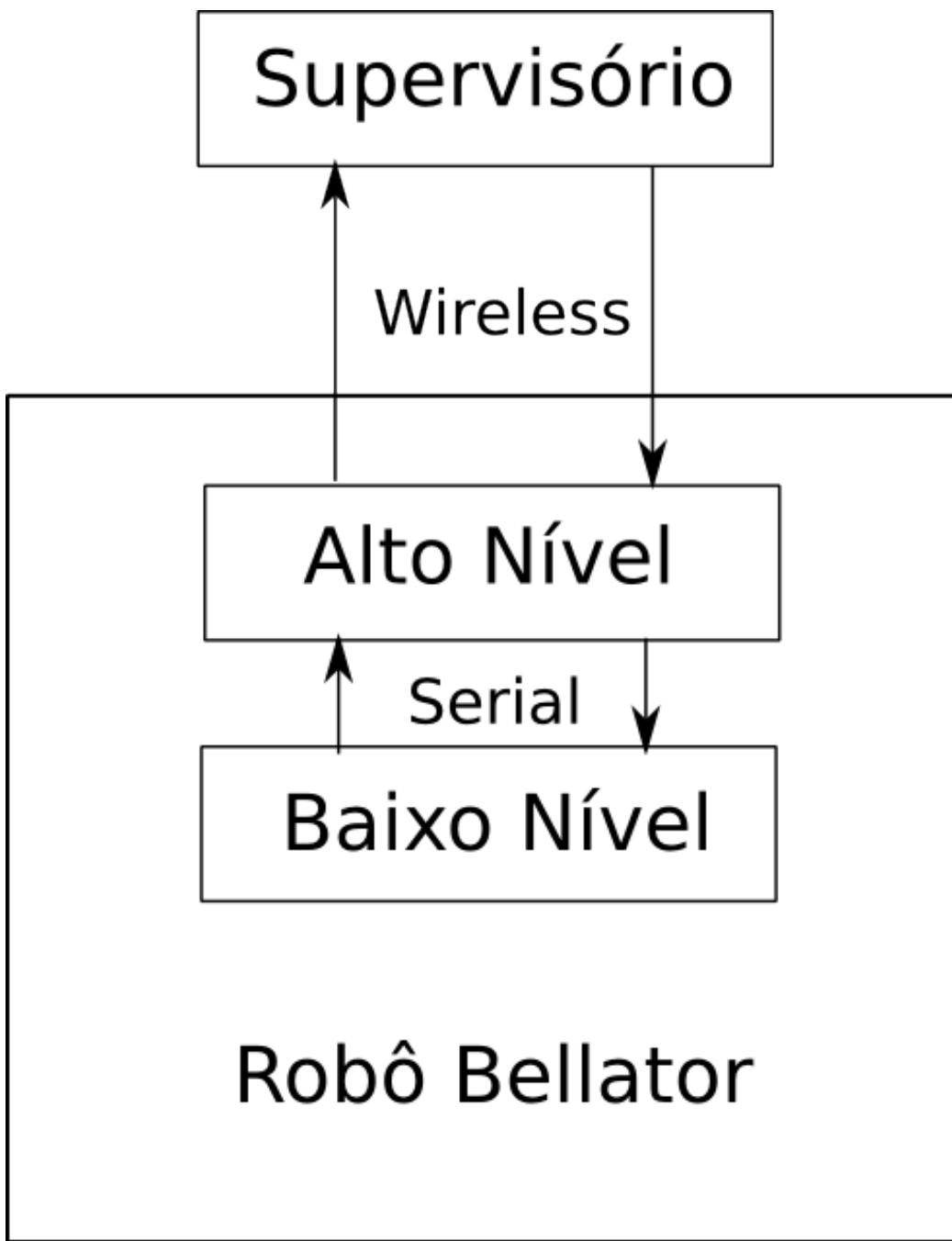


Figura 1: Visão Geral do Sistema

O robô Bellator é constituído das camadas de baixo nível e alto nível. A camada supervisória não faz parte do robô, apenas comunica-se com ele.

Cada uma destas camadas será explicada em maiores detalhes nas seções seguintes.

2.2 Camada Supervisória

2.2.1 Hardware

A camada lógica supervisória, representada na figura 1 não é componente do robô Bellator. Esta camada possui os seguintes elementos:

- Uma Plataforma capaz de executar aplicativos Java, com suporte à vídeo e comunicação Wireless.
- Um Joystick, detalhado na seção 3.2.8

A camada supervisória proporciona ao usuário final uma interface simples de controle da plataforma robótica. Como mencionado acima, é necessária uma plataforma capaz de executar aplicativos Java, visto que o software desenvolvido nesta camada foi escrito em Java e, além disso, é necessário que esta plataforma possua suporte à vídeo, pois o software proporciona realimentação visual para o usuário através de uma interface gráfica. Ainda, a plataforma deve possuir um meio de comunicação Wireless, visto que a camada de alto nível, apresentada na seção 2.3, necessita de uma conexão Wireless com a camada supervisória para o funcionamento do sistema. Os elementos desta camada estão representados no diagrama em blocos a seguir:

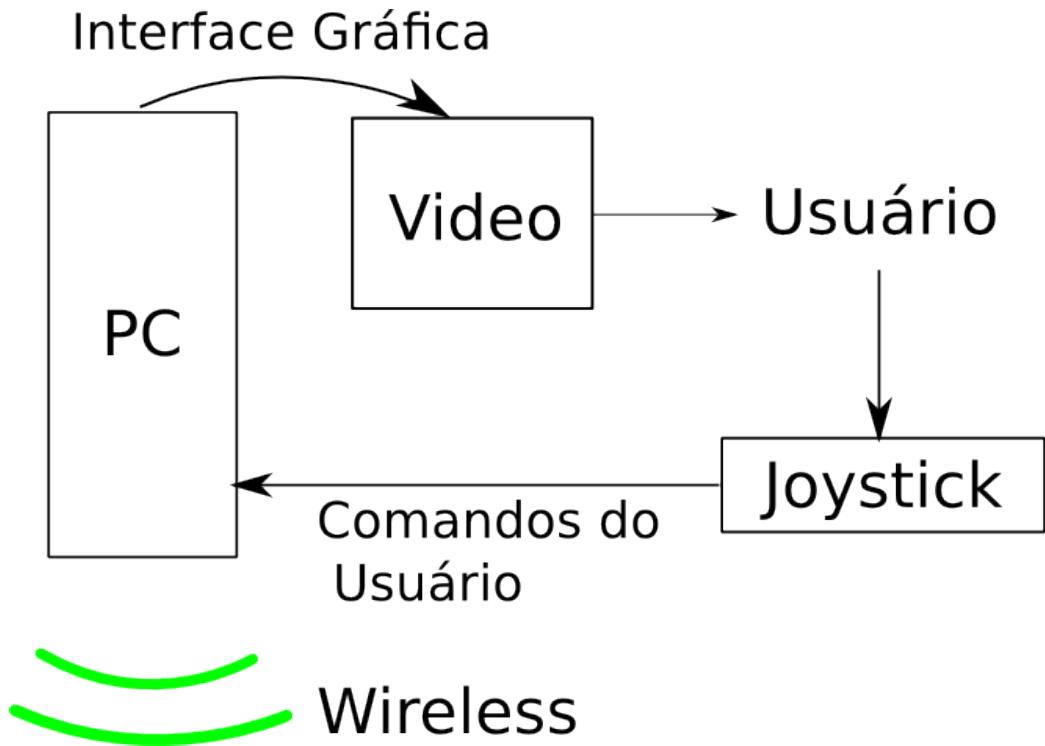


Figura 2: Diagrama em Blocos da Camada Supervisória

2.2.2 Software

O principal componente desta camada é o software Supervisor Remoto. Sua funcionalidade é a de interfaceamento do sistema como um todo com o usuário final. Disponibiliza ao usuário uma consulta aos dados coletados do robô, como imagens capturadas pela webcam e medições de distância realizadas pelos sensores infra-vermelhos. Também capacita o usuário a controlar o robô por meio de um joystick, o qual deve estar acoplado ao computador executando o software supervisor remoto.

Como o software foi desenvolvido em Java, este pode ser executado em plataformas Windows, Linux e Mac.

2.3 Camada de Alto Nível

2.3.1 Hardware

A camada lógica de alto nível, representada na figura 1 é componente do robô Bellator. Os elementos desta camada são:

- PC Embarcado, detalhado na seção 3.2.6
- Uma Webcam, detalhada na seção 3.2.4

Esta camada é fundamental ao sistema, pois deve proporcionar um canal de comunicação entre a camada supervisória e a de baixo nível, bem como prover uma realimentação visual para a camada supervisória por meio de um stream de vídeo. Estas funções não poderiam ser realizadas apenas com os elementos da camada de baixo nível. Os elementos desta camada estão representados no diagrama em blocos a seguir.

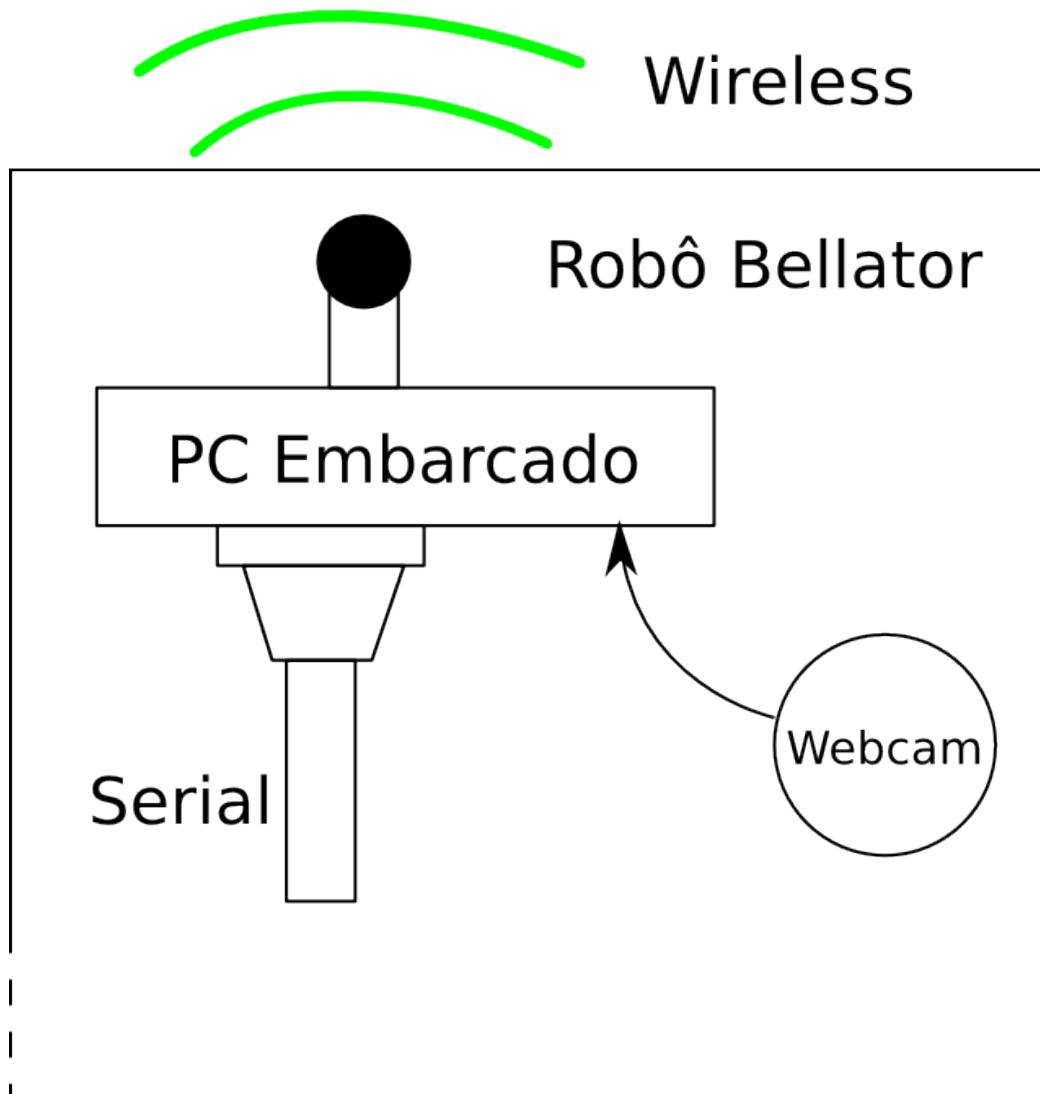


Figura 3: Diagrama em Blocos da Camada de Alto Nível

2.3.2 Software

O software desenvolvido para esta camada tem a função de fazer a comunicação entre as camadas de baixo nível e supervisória. Funciona basicamente como um roteador de mensagens entre estas duas camadas, encaminhando comandos de movimentação advindos da camada supervisória à camada de baixo nível e medições realizadas pelos sensores IR, bem como as imagens capturadas pela webcam, à camada supervisória.

2.4 Camada de Baixo Nível

2.4.1 Hardware

A camada lógica de baixo nível, representada na figura 1, faz parte do robô Bellator. Esta camada possui os seguintes elementos:

- A placa C8051F340DK, detalhada na seção 3.2.1
- Seis sensores infra-vermelho detalhados na seção 3.2.2
- Uma placa para roteamento das leituras dos sensores e dos PWMs para a placa C8051F340DK, descrita em 3.2.3
- Motores, rodas e outros elementos detalhados em 3.2.5

Esta camada deverá realizar a geração de PWMs para o controle dos motores do robô, bem como realizar a leitura dos sensores infra-vermelho. A geração do PWM deve ser controlada por meio de uma comunicação serial com a camada de alto nível. O diagrama em blocos a seguir representa esta camada e seus elementos.

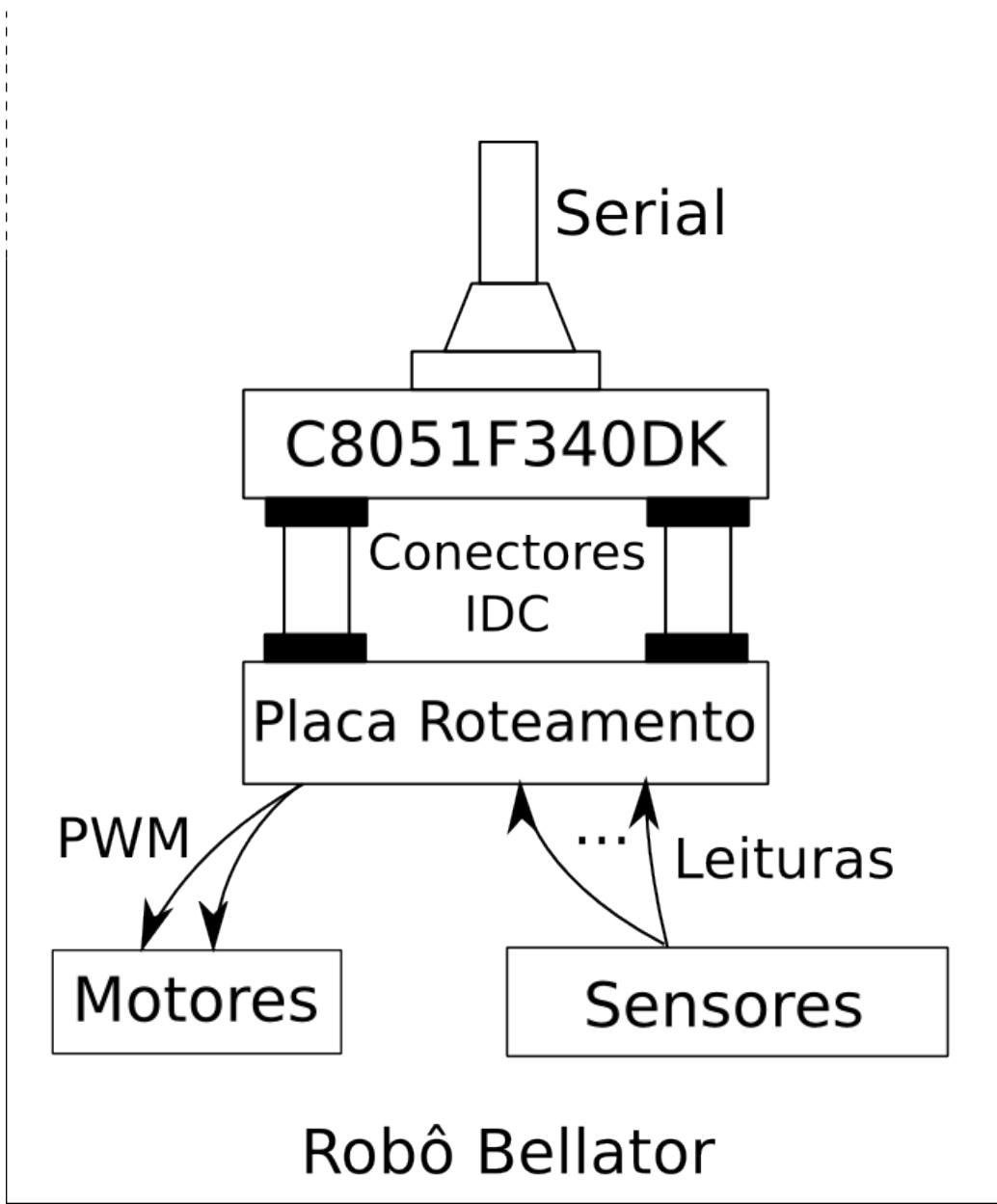


Figura 4: Diagrama em Blocos da Camada de Baixo Nível

2.4.2 Software

O software da camada de baixo nível executa na placa C8051F340DK e é responsável por controlar esta placa e suas funções na camada de baixo nível. A função deste software é geração de PWM para os dois motores do robô,

a varredura dos sensores de distância e envio das informações obtidas destes sensores para a camada de alto nível e, ainda, a interpretação e execução dos comandos recebidos da camada de alto nível.

2.5 Protocolo de Comunicação

O protocolo de comunicação, que foi utilizado por todas as camadas (supervisória, alto nível e baixo nível), consiste em uma especificação de comandos. Uma lista de todos os comandos do protocolo pode ser acessada no anexo A. Todas as mensagens passadas seguem o padrão:

COMANDO VALOR FIM_COMANDO

Onde COMANDO é um byte que representa qual é o comando, VALOR um conjunto de zero ou mais bytes para o valor do COMANDO (dependendo do comando) e FIM_COMANDO é o byte 0xFE, que indica que o comando termina ali.

Dentre os comandos previstos no protocolo, os utilizados foram:

- RODA_LEFT (0xA0): Este comando serve para controlar o sentido de rotação e a largura de PWM do motor que controla a roda esquerda do robô. A mensagem passada segue o padrão:

RODA_LEFT VALOR FIM_COMANDO

VALOR é composto por apenas 1 byte no caso deste comando, onde o bit mais significativo representa o sentido (1 para fazer a roda girar pra frente e 0 para o contrário), e os outros 7 bits correspondem a um valor decimal entre 0 e 15 inclusive, para selecionar um dos 16 níveis de PWM. O valor 0 indicaria o motor parado, enquanto o valor 15 corresponde à velocidade máxima.

- RODA_RIGHT (0xA1): Funcionamento idêntico ao comando anterior, só que para o controle do motor da roda direita (sentido de rotação e largura de PWM).
- STOP (0xFF): Comando para zerar a PWM dos dois motores, não tem nenhum byte de valor, utilizado da seguinte forma:

STOP FIM_COMANDO.

- SENSOR_OPTICO_0 (0x22): Comando utilizado para passar a leitura do sensor óptico 0, já convertida pelo conversor AD. A mensagem contém, então, 3 bytes:

SENSOR_OPTICO_0 VALOR FIM_COMANDO,

onde VALOR é o valor da conversão AD, composto por apenas 1 byte. A mesma estrutura é utilizada para os cinco comandos seguintes.

- SENSOR_OPTICO_1 (0x23): utilizado para o sensor 1.
- SENSOR_OPTICO_2 (0x24): utilizado para o sensor 2.
- SENSOR_OPTICO_3 (0x25): utilizado para o sensor 3.
- SENSOR_OPTICO_4 (0x26): utilizado para o sensor 4.
- SENSOR_OPTICO_5 (0x27): utilizado para o sensor 5.

Os demais comandos descritos no protocolo são para utilização em uma continuação do projeto.

2.6 Bibliotecas Externas

O projeto inclui, basicamente, duas tarefas que não são nativamente executadas pela linguagem Java, a reprodução de vídeo e a interface com o joystick. Para realizá-las, a equipe procurou por bibliotecas em Java que fossem gratuitamente distribuídas e que realizassem estas tarefas.

Para a reprodução de vídeo, o player VLC [8] disponibiliza uma biblioteca para desenvolvedores, a libVLC, distribuída sob a licença GNU General Public License v2 [6]. Como esta biblioteca é escrita em linguagem C++, a equipe utilizou um wrapper para Java, o vlcj [9] que é distribuído sob a licença GNU General Public License v3 [7]. Este wrapper tem determinadas funções que permitem ao desenvolvedor tratar um stream, seja ele de áudio ou vídeo, com as funcionalidades implementadas na libVLC. Neste projeto foram utilizadas somente as funções de conexão de stream de vídeo, como abrir uma nova conexão, tocar, pausar e parar o vídeo. A interface de vídeo com o usuário pode ser feita utilizando tanto a Java AWT Window como o Swing JFrame.

O vlcj necessita da biblioteca JNA [10] para acessar a libVLC nativa do sistema onde o código esteja sendo executado. O JNA é distribuído sob a licença GNU Lesser General Public License v3 [13].

Para a interface com o joystick, foi utilizada a biblioteca JInput [11]. A JInput é distribuida gratuitamente sob a licença BSD [12]. Esta tem como funcionalidade o reconhecimento e a captura dos comandos de interfaces de controle como teclado, mouse e joysticks, porém neste projeto foi utilizada somente esta última. Para captura dos comandos, a biblioteca disponibiliza a técnica de pooling, sendo este retorno do controlador analógico dividido em dois valores: componente do eixo das abscissas e das ordenadas. Estes valores estão situados no intervalo de [-1;1], obtendo-se valores positivos com comandos acima do eixo de origem para o caso das ordenadas e a direita do mesmo para o eixo das abscissas.

3 Projeto

3.1 Projeto do Software Supervisor Remoto

O sistema supervisor remoto é um software desenvolvido em linguagem Java, o qual basicamente tem como objetivo interfacear o sistema Bellator com o usuário. O supervisor mostra a leitura dos sensores, juntamente com as imagens captadas pela webcam, para guiar o usuário no controle remoto do robô, o qual é realizado através do joystick.

A escolha da linguagem utilizada no desenvolvimento do software foi baseada em alguns requisitos específicos. Um destes foi o suporte para tratamento de stream de vídeo, para que fosse possível a agregação do mesmo à interface do software. Outro foi o de suporte para captura de comandos de um joystick, para que o usuário possa utilizá-lo para controlar a movimentação do robô. A linguagem escolhida deve também ter capacidade de tratamento de sockets, com o fim de concretizar a comunicação wireless de troca de dados com o sistema (tais como enviar comandos do joystick e receber dados dos sensores). Como Java atende a todos estes requisitos, se tornou a opção mais viável para o desenvolvimento do sistema supervisor remoto.

Para desenvolver esta etapa fundamental foi realizado um projeto do software, o qual é composto por um levantamento de requisitos, casos de uso, e um diagrama de classes. Estes serão abordados nas seções seguintes.

3.1.1 Levantamento de Requisitos

Para iniciar o desenvolvimento de um software é necessário primeiramente saber o que o software deve ser capaz de fazer, quais suas funcionalidades e restrições. Portanto foi desenvolvido um levantamento de requisitos para o sistema supervisor remoto.

Requisitos Funcionais:

- O software deverá mostrar a imagem capturada pela webcam na tela para o usuário.
- O software deverá mostrar os dados dos sensores (adquiridos via interface serial) na tela para o usuário.
- O software deverá possibilitar ao usuário controlar a movimentação do robô remotamente através de um joystick.
- O software deverá possibilitar ao usuário a configuração da conexão de dados com o robô para envio de comandos do joystick e recebimento de dados capturados pelos sensores.
- O software deverá possibilitar ao usuário a configuração da conexão do stream de vídeo.
- O software deverá possibilitar ao usuário o acesso através da interface gráfica ao manual do usuário.

Requisitos Não-Funcionais:

- O software poderá ser executado em plataformas Windows, Linux e Mac.
- O delay do stream de vídeo não deverá ultrapassar os 1500ms.
- O software deverá ser desenvolvido em Java, utilizando o paradigma orientado a objetos.

3.1.2 Estudo de Casos de Uso

Feito o levantamento de requisitos, o próximo passo do projeto do software supervisor remoto foi o estudo dos casos de uso do mesmo, o qual foi baseado nos requisitos levantados do sistema. Este estudo mostrou que o software supervisor remoto é composto por seis principais casos de uso mostrados na figura 5 .

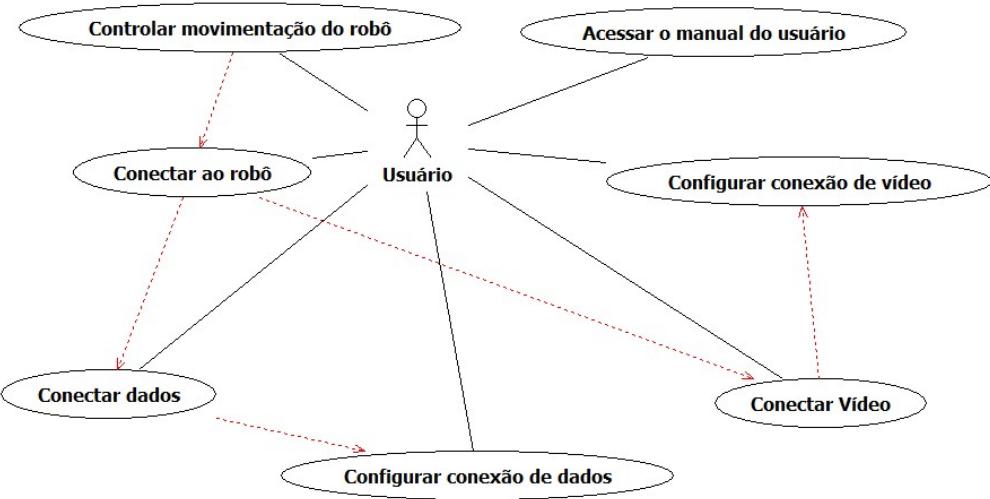


Figura 5: Diagrama de Casos de Uso.

Para que o usuário possa interagir com o robô, primeiramente é necessário que o usuário conecte o sistema supervisor remoto ao robô. Para isso é necessário que exista uma conexão de rede entre o PC embarcado e o PC supervisor remoto. Tendo isto, é necessário fornecer o IP e a porta da conexão do socket de dados, respeitando o formato (IP:Porta), para que os dados dos sensores possam ser mostrados na tela para o usuário. Também é necessário fornecer ao sistema a URL do stream de vídeo, a qual tem o formato (<http://IP:Porta/robo.mjpeg>), para que o usuário possa ter à disposição as imagens capturadas pela webcam do robô. Caso não exista conexão de rede entre o PC embarcado e o supervisor remoto, a interação com o robô torna-se indisponível.

Com estas configurações, o usuário pode visualizar os dados dos sensores e o vídeo capturado pela câmera, porém ainda não é possível o controle remoto do mesmo. Para isto, além de todos os requisitos anteriormente citados, é preciso ter um joystick conectado ao PC supervisor remoto, para que o usuário possa utilizá-lo no controle de movimentação do robô. Caso o sistema não encontre um joystick, o usuário é informado pela própria interface e o controle do robô fica indisponível.

3.1.3 Diagrama de Classes

Para guiar o desenvolvimento do software, foi construído um diagrama de classes inicial seguindo o padrão UML [14], sendo este editado e atualizado

ao longo da fase implementação. O diagrama de classes final do sistema pode ser observado na figura 6. Nota-se que foram previstas seis classes principais para o sistema, sendo a ControlePrincipal responsável por gerenciar todos os componentes utilizados no sistema (comunicação de dados, protocolo de comunicação, stream de Vídeo, controle do robô via Joypad e interface com o usuário), e as demais responsáveis por um componente cada.

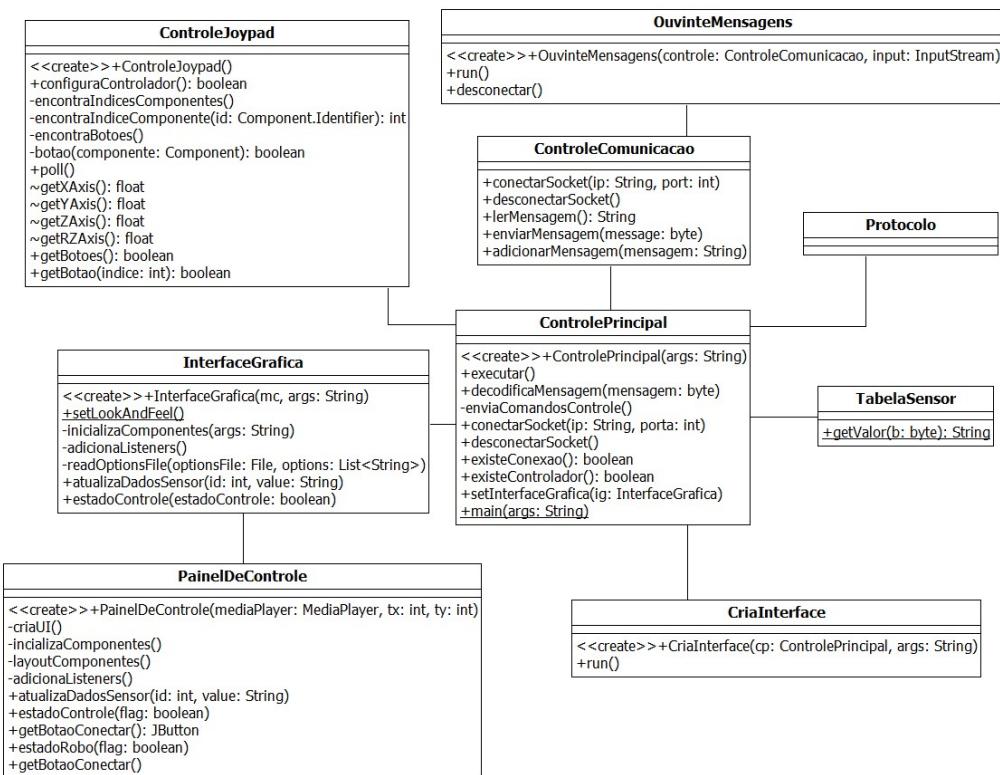


Figura 6: Diagrama de Classes.

Os detalhes de implementação do software serão abordados de maneira aprofundada na seção 4.2.

3.2 Projeto do Hardware

3.2.1 C8051F340DK

O C8051F340 é uma unidade micro controladora (MCU) equipada com um processador da família 8051 e vários dispositivos periféricos dispostos em uma

placa de circuito impresso. Esta unidade faz parte do robô Bellator e possui as seguintes especificações [1]:

- Conversor ADC 10 bits de até 200ksps (amostras por segundo)
- Dois comparadores
- Brown-out Reset e Power-on Reset
- Tensão de Referência interna
- Porta USB 2.0
- Duas interfaces seriais (UART) e uma interface SPI
- Fonte de Alimentação de 2.7 até 5.25V regulada internamente
- Micro-processador 8051 de até 48 MIPS
- 4352 Bytes de memória RAM
- 40 Portas I/O
- 4 Timers de 16 bits
- Seleção de Clock interno de alta ou baixa velocidade ou clock externo

Um diagrama em blocos do kit, retirado do datasheet, pode ser visto abaixo:

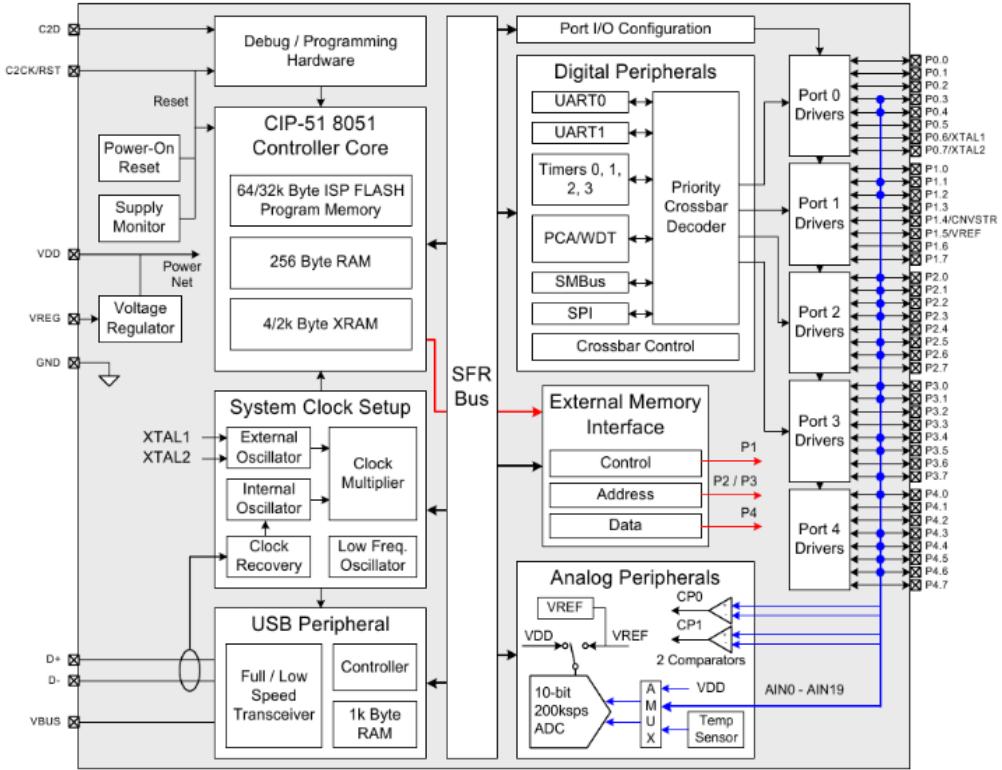


Figura 7: Diagrama em blocos do kit C8051F340DK. Fonte: datasheet.

3.2.2 Sensor IR 2Y0A02F98

Com o objetivo de auxiliar a navegação do robô e fazer varreduras do ambiente, foram instalados seis sensores analógicos de distância por infravermelho modelo 2Y0A02F98 da Sharp, dispostos uniformemente nas laterais da plataforma Bellator. Este modelo mede distâncias no intervalo de 20 a 150 centímetros [3], sendo que os valores de tensão de resposta do sensor seguem a curva mostrada na figura 8 .

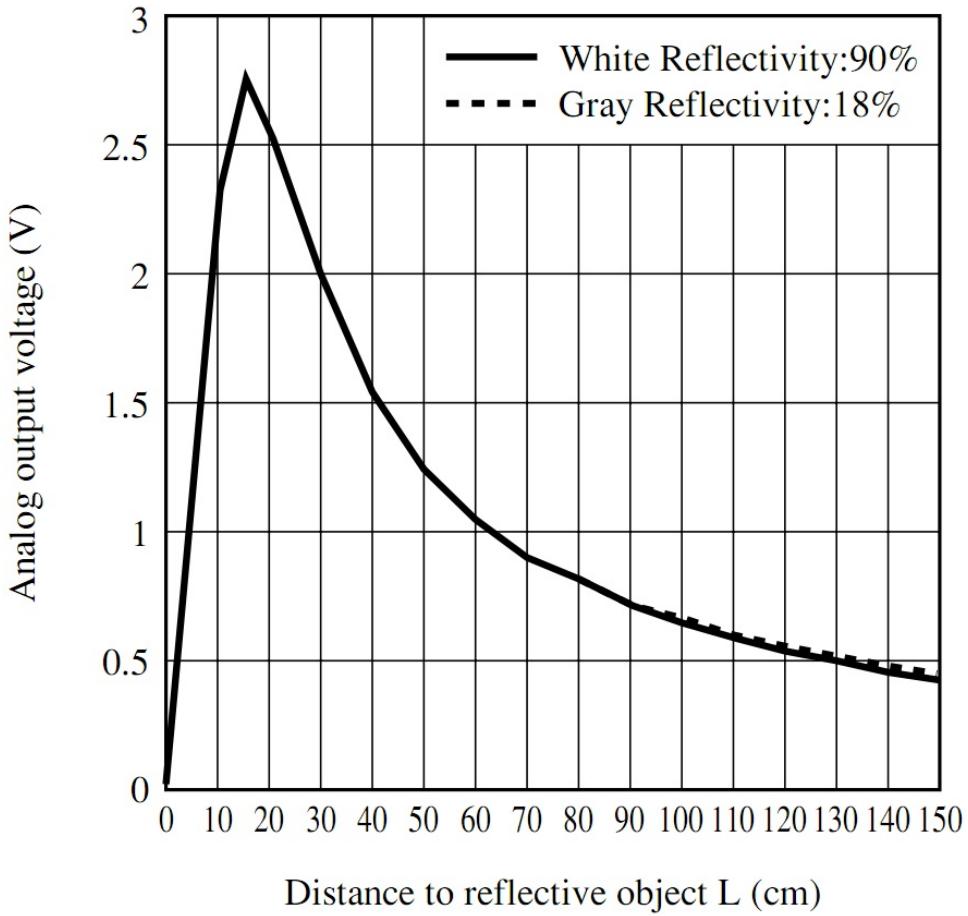


Figura 8: Curva de resposta do sensor de distância. Fonte: datasheet.

Pode-se observar que o modelo é pouco influenciado pelas cores dos objetos refletidos, isto devido ao método de medição baseado em triangulação [3]. O sensor possui uma tensão de alimentação recomendada na faixa de 4.5 a 5.5V, a qual não é atendida pela plataforma utilizada [1], sendo necessária utilização de alimentação especial. Esta alimentação é realizada através de um circuito regulador de tensão, o qual utiliza alimentação da própria bateria acoplada ao robô. O cálculo dos valores dos resistores foram baseados na equação 1.

$$V_{OUT} = 1,25V(1 + R_2/R_1) \quad (1)$$

O diagrama esquemático do regulador de tensão é mostrado na figura 9.

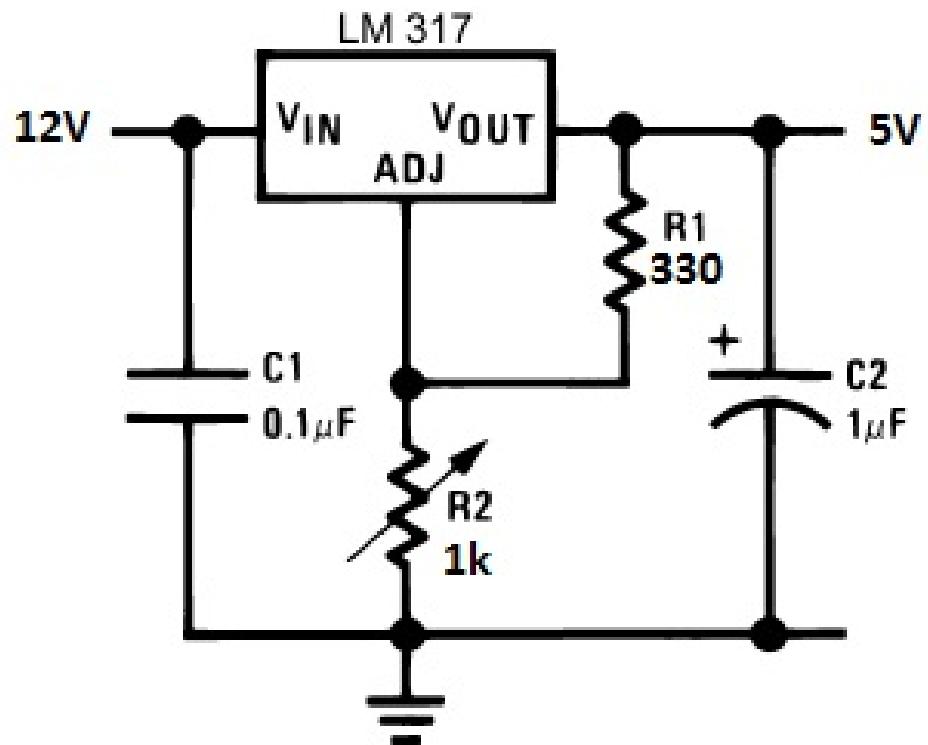


Figura 9: Diagrama esquemático do regulador de tensão dos sensores de distância.

As respectivas dimensões do sensor IR, em milímetros, são mostradas na figura 10

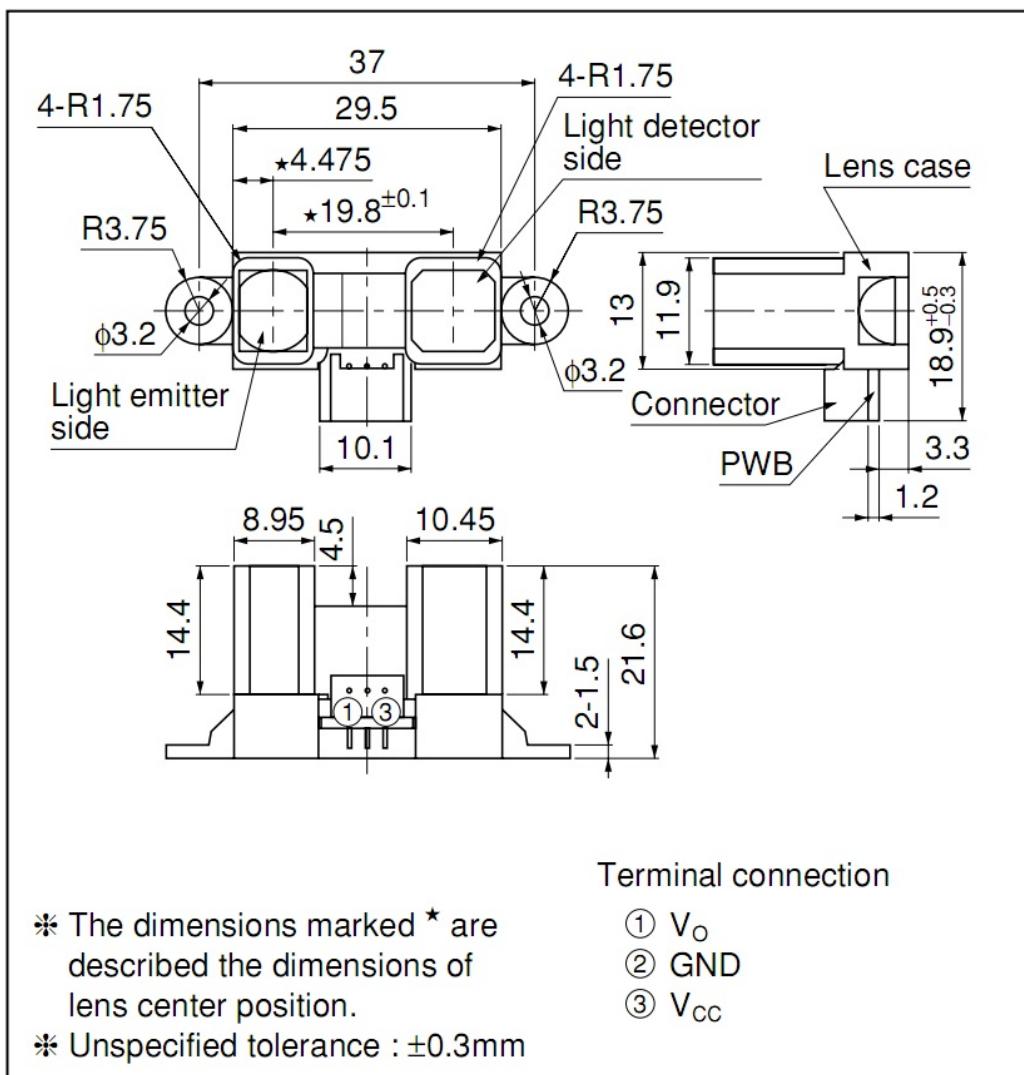


Figura 10: Dimensões do sensor GP2Y0A02F98YK. Fonte: datasheet.

Para agregar os sensores à plataforma, primeiramente foi levantada a curva de resposta dos mesmos. Com estes dados, foi feita uma aproximação polinomial da curva de resposta e, com a equação resultante 2, foram calculados 256 pontos entre o intervalo de operação. Esta tabela resultante, de formato tensão(V) X distância(cm), foi armazenada no nível supervisor remoto para ser utilizada para obtenção das respostas dos sensores (distância

em centímetros).

$$V_{medido} = 2,01091E - 8x^4 - 8,7740E - 6x^3 + 0,0014x^2 - 0,1152x + 4,3134 \quad (2)$$

O modelo em questão é adequado ao projeto pois, como sua principal finalidade é a de ajudar a navegação do robô em ambientes fechados, sua faixa de resposta de 20 a 150 centímetros é suficiente para detecção de objetos. Contudo, há a possibilidade de num projeto futuro serem acrescentados outros tipos de sensores mais precisos voltados à medição de distâncias menores.

3.2.3 Placa de Roteamento

A placa de roteamento foi desenvolvida pela equipe visando proporcionar a alimentação de 5 Volts necessária para os sensores infra-vermelho, descritos na seção 3.2.2, rotear cada leitura dos sensores para o PORT2 da C8051F340DK e fazer um buffer para os sinais de PWM gerados pela C8051F340DK.

A figura a seguir é um diagrama do circuito da placa de roteamento.

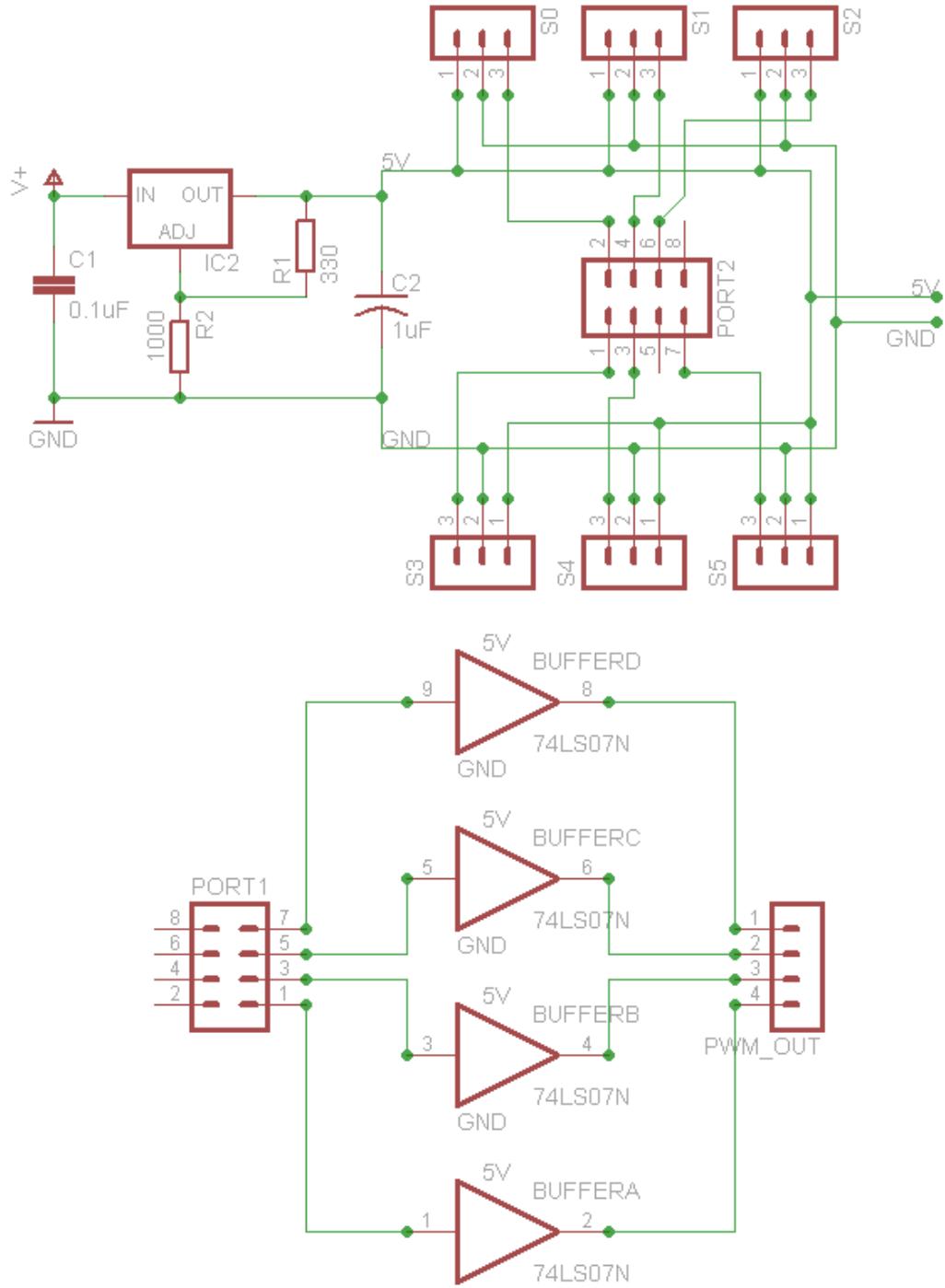


Figura 11: Diagrama da Placa de Roteamento

3.2.4 Webcam

O modelo da webcam utilizada no robô Bellator é Genius iLook 316, que possui as seguintes especificações :

- Resolução: VGA 640x480 pixels.
- Taxa máxima de frames por segundo: 15, na resolução utilizada.

Mais especificações sobre a série iLook 300 da Genius podem ser encontradas no site oficial da empresa, incluso nas referências bibliográficas. [2]

A escolha do modelo de webcam já tinha sido realizada, sendo que a mesma atende os requisitos do projeto.

3.2.5 Robô Bellator

O robô Bellator possui duas rodas de tração e uma roda de apoio. As rodas de tração estão nas laterais da parte traseira do robô e possuem diâmetro de 20 centímetros e espessura de aproximadamente 4 centímetros. Ambas possuem um encoder de quadratura acoplado, o qual tem resolução de 1800 pulsos por volta. A roda de apoio está no centro da parte dianteira do robô e possui diâmetro de aproximadamente 6 centímetros e espessura de 2 centímetros. Todas as rodas são da marca Schioppa. Os outros componentes do robô estão listados a seguir:

- 2 Motores Bosch FPG 12V
- Bateria Unybatt 12V-7.2 Ampére-hora
- Duas pontes H L 298

Uma imagem do robô, parcialmente montado, pode ser visualizada a seguir:

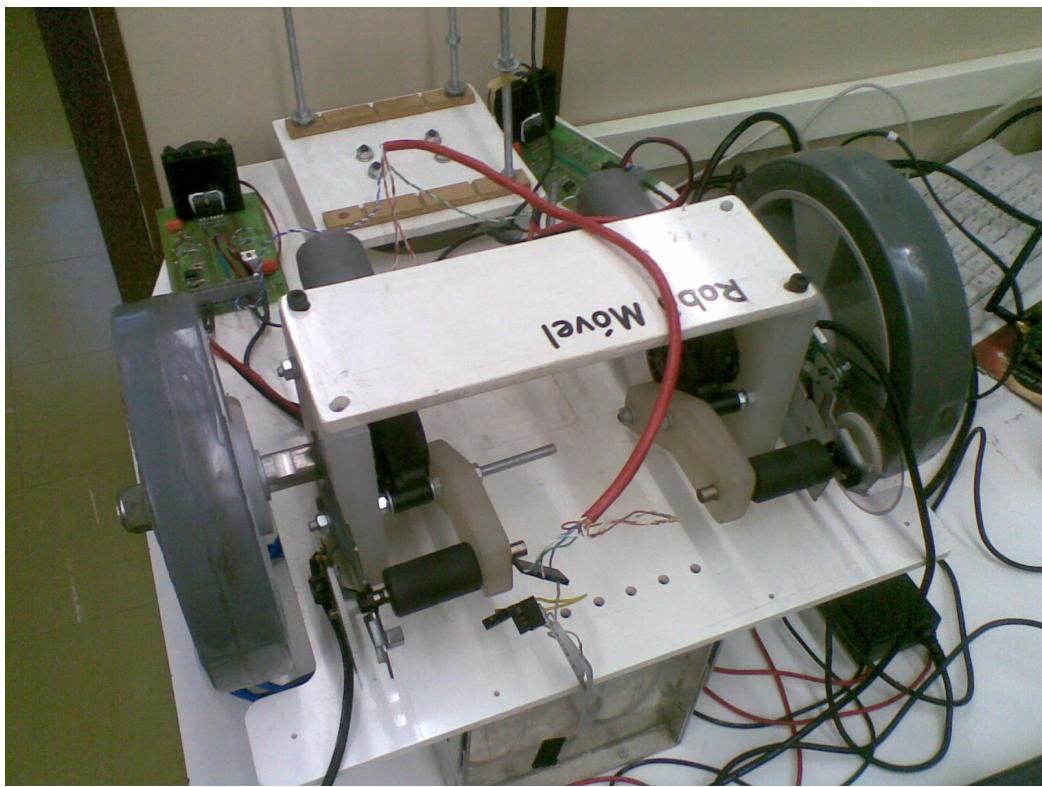


Figura 12: Foto do robô Bellator.

3.2.6 PC embarcado

O PC embarcado é uma VIA EPIA M-Series Mini-ITX Mainboard, uma plataforma 32bits ultra compacta projetada especialmente para aplicações digitais embarcadas. A placa possui as dimensões de 17 por 17 centímetros e vários dispositivos integrados. Alguns deles estão listados a seguir:

- Processador VIA C3/Eden EBGA
- Slot DDR226DIMM
- Duas portas IDE ATA /133/100
- Um slot PCI
- Duas portas USB 2.0
- Uma porta ethernet 10/100

Uma imagem descritiva com os principais elementos do PC embarcado, retirada do datasheet da mesma [4], pode ser visualizada a seguir:

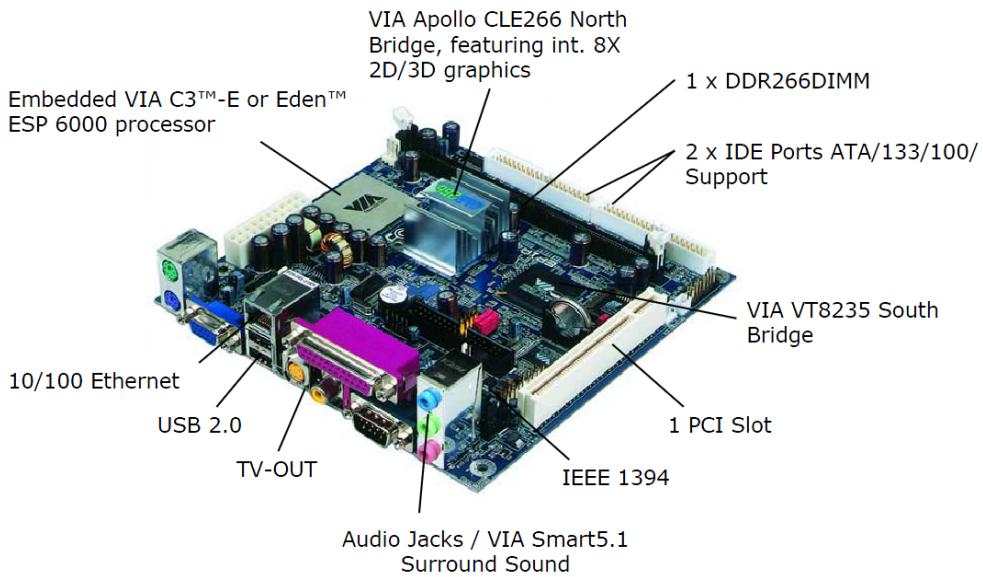


Figura 13: VIA EPIA M-Series MiniITX Mainboard. Fonte: Datasheet.

Além do que está representado na 13, o PC embarcado está equipado com uma memória RAM PC133 de 128mb e uma placa de rede Wireless Edimax PCI-LAN EW-7128G [5] .

3.2.7 Interconexão do Hardware

Os elementos de hardware descritos até agora devem ser conectados de forma correta para o funcionamento da plataforma robótica Bellator. Esta seção visa descrever como estas conexões devem ser feitas. Cabe aqui lembrar que, como descrito nas seções 2.3.1 e 2.4.1 o robô Bellator é composto por todos os elementos da camada de baixo e alto nível, ou seja:

- A placa C8051F340DK, detalhada na seção 3.2.1
- Seis sensores infra-vermelho detalhados em 3.2.2
- Uma placa para roteamento das leituras dos sensores e dos PWMs para a placa C8051F340DK
- Motores, rodas e outros elementos detalhados em 3.2.5

- PC Embarcado, detalhado na seção 3.2.6
- Uma Webcam, detalhada na seção 3.2.4

As figuras a seguir, explicadas nos parágrafos seguintes, servem como instrução para o posicionamento destes elementos:

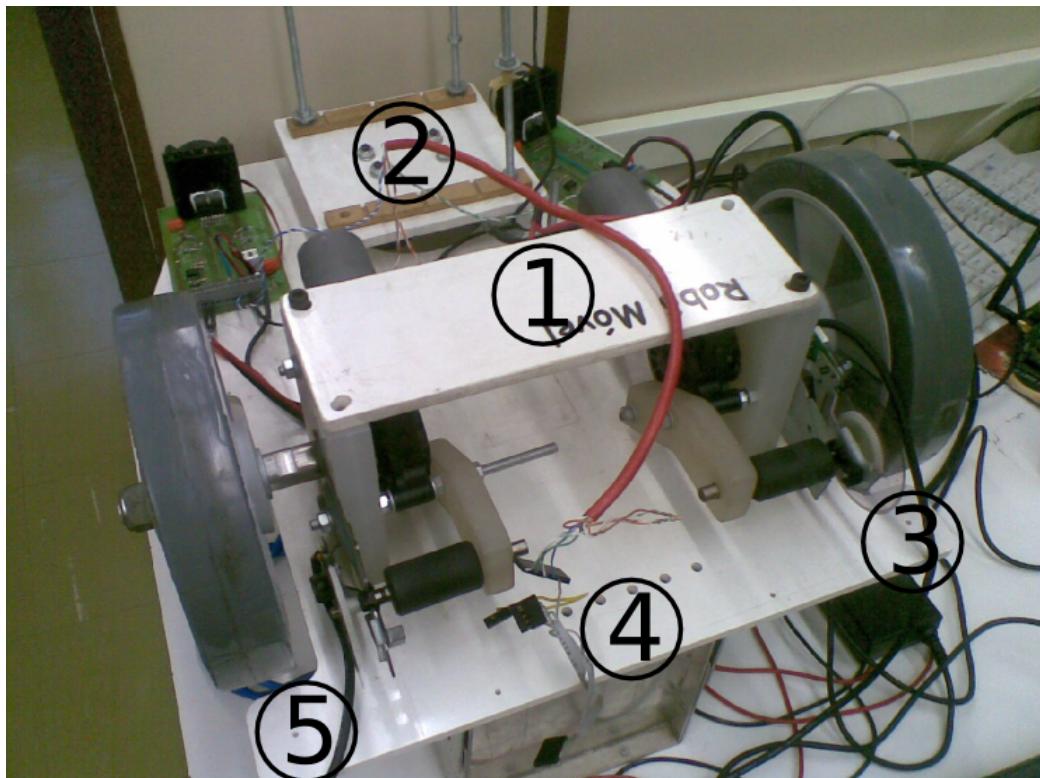


Figura 14: Posicionamento dos Elementos do Robô - Parte 1

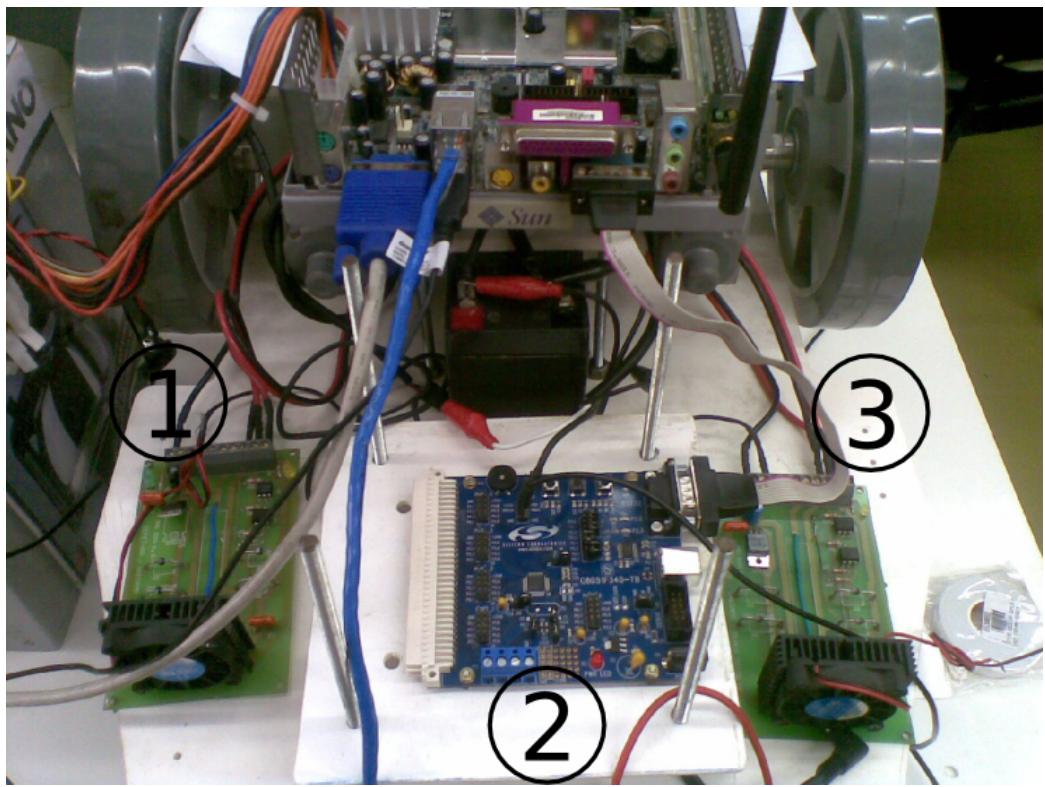


Figura 15: Posicionamento dos Elementos do Robô - Parte 2

Na figura 14, que é uma foto editada do robô, antes da montagem final do mesmo, os números correspondem aos seguintes elementos:

1. Nesta plataforma elevada deverá ser colocado o PC embarcado. Abaixo desta plataforma encontram-se os motores e um espaço (em baixo relevo na superfície do robô) onde a bateria deve ser posicionada.
2. Nesta plataforma, na parte posterior do robô, devem ser colocados a placa C8051F340DK e, logo abaixo, a placa de roteamento.
3. Posição do Sensor 1
4. Posição do Sensor 2
5. Posição do Sensor 3

Os sensores 1 e 3 devem estar inclinados de 45º em sentidos opostos de forma a apontar, respectivamente, para o noroeste e nordeste do robô, onde

o norte do robô é o número 4 na figura 14. A Webcam deverá ser posicionada acima do PC embarcado e conectada neste via USB.

Na figura 15, retirada durante a montagem do robô, os números correspondem aos seguintes elementos:

1. Posição do Sensor 4
2. Posição do Sensor 5
3. Posição do Sensor 6

Os sensores e os sinais de PWM devem ser conectados à placa de roteamento de acordo com o diagrama da placa de roteamento, apresentado na seção 3.2.3. De acordo com a figura 11, cada sensor deve ser conectado da seguinte forma: Sensor 1 em S0, Sensor 2 em S1, até Sensor 6 em S5. Nota-se que o número 1 representa a alimentação de 5V do sensor, que corresponde ao cabo vermelho do conector deste. Os conectores DCI devem ligar o PORT1 da placa de roteamento ao PORT1 da placa C8051F340DK e, analogamente, o PORT2 da placa de roteamento ao PORT2 da C8051F340DK. Os sinais de PWM 1 e 2 devem ser conectados à ponte H da direita do robô e os sinais 3 e 4 na da esquerda, a partir da perspectiva vista na figura 15.

Além disso, a porta serial da placa C8051F340DK deve ser conectada ao pc embarcado e, finalmente, a alimentação da placa de roteamento, da C8051F340DK, das pontes H, motores e da fonte do pc embarcado devem ser conectadas nos pólos da bateria.

3.2.8 Joystick

O joystick utilizado no projeto é um controle de Sony Playstation 2. Primeiramente a escolha por utilizar um joystick foi devido a geração de sinal analógico de saída, o que o torna interessante para controle da velocidade de movimentação do robô. Dentre as opções de controladores analógicos, esta escolha foi feita devido à maior precisão do sinal analógico gerado pelo controle em questão, ou seja, este possui uma maior resolução. Outro ponto forte é a presença de dois controladores analógicos, sendo um voltado ao movimento do robô e o outro para possível controle da câmera num projeto futuro.

A conexão deste controlador é padrão da Sony PS2, sendo portanto utilizado um adaptador deste padrão para USB, com o intuito de conectá-lo ao computador. Uma imagem do joystick, juntamente com seu adaptador para USB, pode ser visualizada na figura 16.



Figura 16: Joystick de Sony Playstation 2. Fonte <http://www.easytechnology.gr/images/PS2_sony_ps2_controller.jpg>

Agregando este controlador à biblioteca JInput (veja 2.6), os valores retornados pela mesma quando aplicado um comando em cada eixo pertencem ao intervalo [-1;1], sendo o valor positivo para cima da origem no eixo das ordenadas e para esquerda no eixo das abscissas.

Os comandos de movimento do robô são realizados utilizando o controle analógico direito do joystick. Se o comando possuir uma componente no eixo Y este se move proporcionalmente a essa componente em módulo e com mesmo sentido. Se contiver componente no eixo X o movimento é de rotação (sentido horário para componente positiva, caso contrário sentido anti-horário).

4 Execução

4.1 Estado do Sistema

Como já mencionado anteriormente, este projeto é a continuação de trabalho que vem sendo desenvolvido no robô Bellator, e no qual várias outras pessoas já trabalharam. Assim, convém descrever quais eram as características do robô quando o projeto foi iniciado pela equipe. Na camada de alto nível, o robô possuía um PC Linux, que não foi alterado pela equipe, e na camada de baixo nível possuía uma placa com processador AT89C52 em uma placa P51, com um código funcional, ao qual a equipe teve acesso.

A equipe analisou o andamento do projeto entregue através de discussão com o responsável anterior e pela análise dos programas já existentes, e a partir disto, concluiu que o mesmo possuía as seguintes características:

- Geração de PWM a partir do AT89C52 para acionamento dos motores, controlável com comandos enviados através da porta serial.
- Recepção da imagem da Webcam pelo PC Linux, roteamento e acesso do stream por uma conexão Wireless funcional.
- Todas estas funcionalidades foram testadas em módulos separados apenas, e não como um todo.
- Um protocolo de comunicação, que definia os vários comandos possíveis.
- Não havia, ainda, leitura de nenhum sensor infravermelho.

O software executando no PC Linux embarcado, como mencionado, não foi alterado e já fazia as operações necessárias para o sistema, descritas na seção visão geral do sistema. Também foi entregue um software, ainda no início do seu desenvolvimento, que capturava comandos do joystick. A visualização do stream de vídeo da webcam era realizada através do VLC player [8]. Este software não foi utilizado pela equipe, a mesma optou por implementar um novo programa em Java, como mencionado na seção 3.1, visando um programa mais versátil e independente de plataforma.

4.2 Software Supervisor Remoto

O software supervisor remoto, como citado anteriormente, é responsável pelo gerenciamento do sistema como um todo e da interface com o usuário. A interface do programa pode ser observada na figura 17.

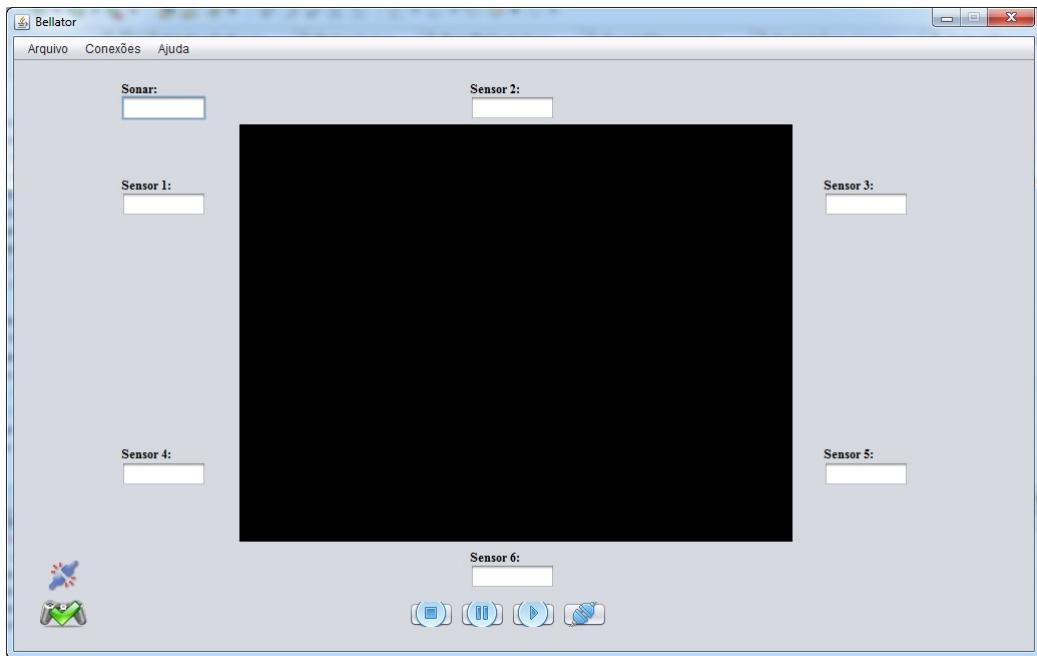


Figura 17: Interface do software Supervisor Remoto.

Este pode ser dividido em basicamente cinco módulos: Util, Principal, Controlador, Comunicação e Interface Gráfica, os quais serão detalhados nas seções a seguir.

4.2.1 Util

O núcleo Util consiste basicamente em dados a serem consultados pelo software durante sua execução, dados que correspondem ao protocolo (ver 2.5) de comunicação utilizado na troca de mensagens entre o controle embarcado (alto nível) e o supervisor, e a tabela de respostas dos sensores IR acoplados ao robô. Toda mensagem entre as camadas de alto nível e supervisória consiste em um vetor de bytes e deve respeitar este protocolo, que determina a formatação da mesma para envio/recebimento.

4.2.2 Principal

O módulo Principal é responsável pela coordenação da execução do software, ou seja, é este núcleo que faz a agregação de todos os módulos para formar um só sistema. A execução do software inicia com a chamada do método executar da classe ControlePrincipal. Este módulo consiste na thread que executa o laço principal do programa, o qual é responsável pela

leitura, decodificação das mensagens recebidas pelo módulo de Comunicação (ver refsec:comunicacao); atualização dos dados na interface gráfica e; encaminhamento dos comandos do joystick, capturados através do módulo Controlador (ver 4.2.3), para a camada de alto nível utilizando novamente o núcleo de Comunicação do software. O diagrama da figura 18 ilustra o funcionamento deste módulo.

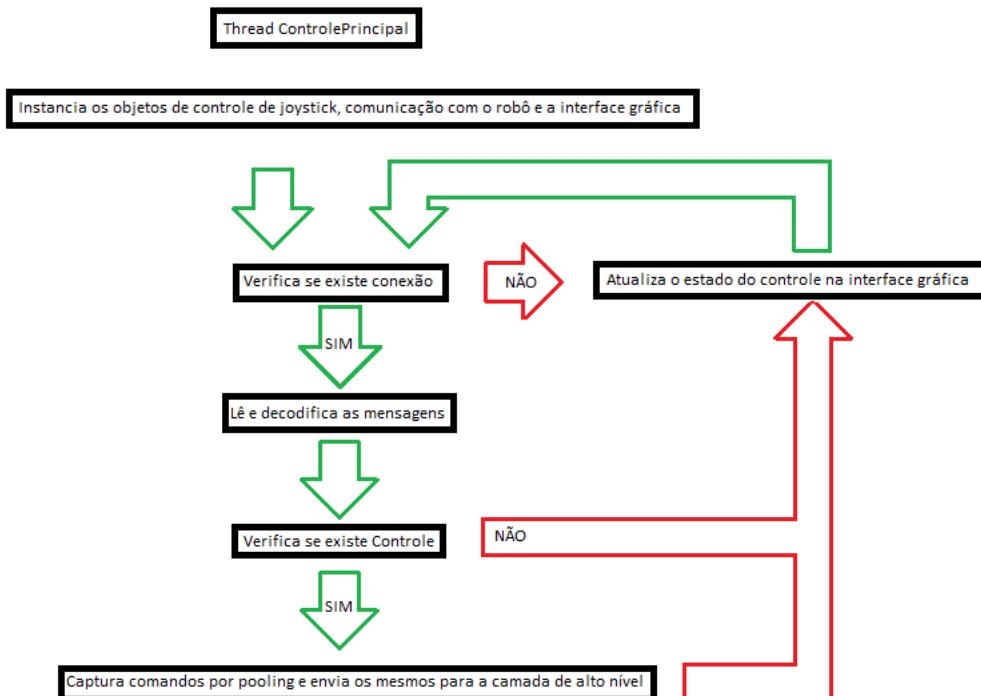


Figura 18: Funcionamento do módulo Controle Principal.

Para envio dos comandos realizados pelo usuário, primeiramente o software captura valores do joystick através de polling e os converte em valores inteiros no intervalo [0, 15] que são interpretados pelo robô. Feita a transformação, é necessária uma lógica de controle do robô através do joystick, já que para movimentação do robô é preciso mandar uma mensagem para cada roda. Esta lógica é bem simples e é detalhada a seguir:

- Se o comando só possui componente no eixo Y, as duas rodas recebem uma força proporcional à aplicada ao joystick;
- Se o comando, além de possuir componente no eixo Y, também possuir no eixo X, uma das rodas tem sua força diminuída de acordo com

o sentido da componente em X, enquanto a outra mantém sua força aplicada;

- Se o comando somente possuir componente no eixo X, é aplicada em ambas as rodas metade da força total, porém em sentidos opostos para realizar o movimento de rotação.

A codificação destas mensagens para envio, é baseada no protocolo (ver 2.5) e feita da seguinte maneira:

- mensagem[byte 0]: Indicador para qual roda é destinada a mensagem (RODA_RIGHT ou RODA_LEFT);
- mensagem[byte 1]: Força da roda (0 a 15). Se o movimento da roda é no sentido oposto soma-se MASCARA_SENTIDO ao byte;
- mensagem[byte 2]: FIM_COMMANDO;
- mensagem[byte 3]: Indicador de nova linha (\n);

A decodificação das mensagens recebidas, sendo no caso dados dos sensores acoplados ao robô, é baseada no protocolo e feita da seguinte forma:

- Faz a seleção baseada no protocolo de qual sensor atualizar através do byte 0 da mensagem;
- Consulta a tabela dos sensores através do byte 1 da mensagem para aquisição da distância medida;
- Atualiza o campo do respectivo sensor na interface gráfica com o valor medido;

4.2.3 Controlador

O módulo do Controlador corresponde à interface com o joystick usado para controlar o robô. É este módulo que a Principal utiliza para os métodos de verificação da existência de um joystick no sistema e captura de seus comandos feitos pelo usuário, sendo este último realizado por pooling. A grande maioria dos métodos implementados neste módulo são baseados no uso da biblioteca JInput, citada na seção 2.6.

4.2.4 Comunicação

O módulo de Comunicação consiste no controle da conexão por socket do supervisor remoto com o sistema de alto nível, sendo assim responsável pelo envio de comandos do joystick e recebimento de medidas dos sensores. Primeiramente esse tenta estabelecer uma conexão com o PC embarcado, sendo que se obtiver sucesso inicializa uma thread que faz o papel de ouvinte de mensagens vindas da camada de alto nível e envia eventuais comandos do joystick pela conexão. O diagrama mostrado na figura 19 mostra o funcionamento deste módulo.

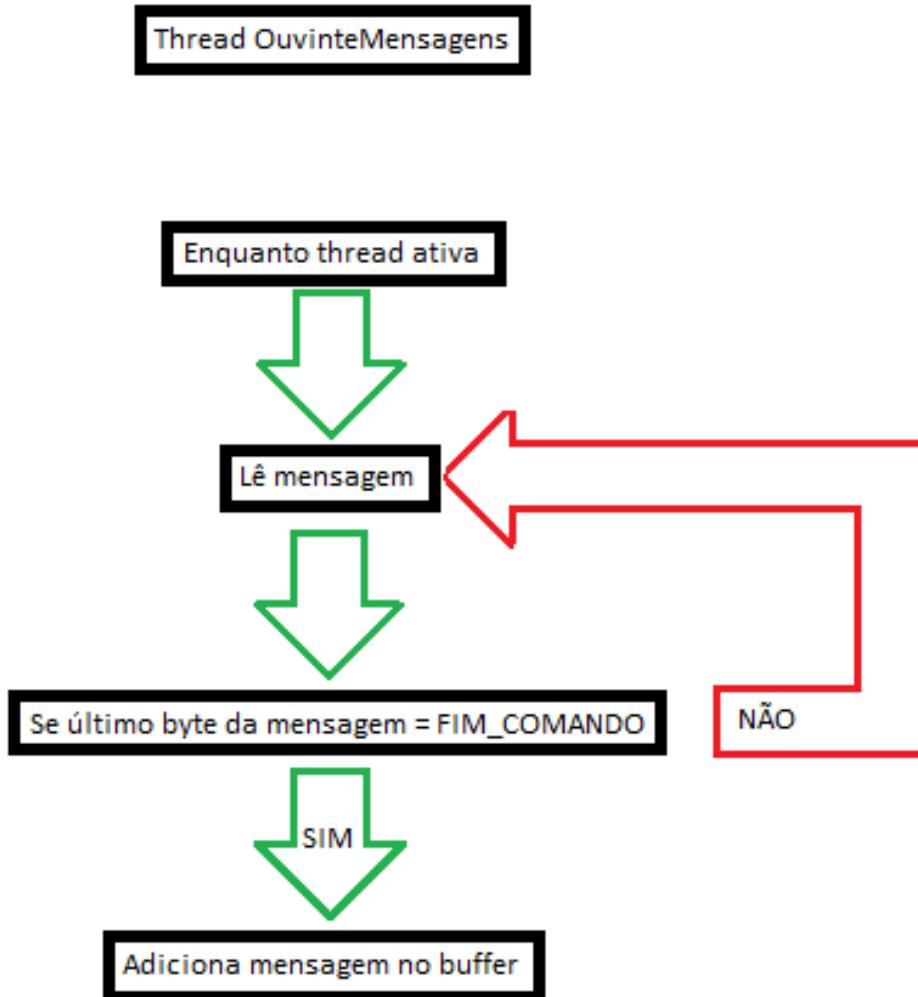


Figura 19: Funcionamento do módulo Comunicação.

4.2.5 Interface Gráfica

O módulo da Interface Gráfica nada mais é do que a interface gráfica do software para a visualização dos dados pelo usuário. Essa consiste numa thread que é responsável por capturar e executar as instruções feitas pelo usuário, tais como configuração do stream de vídeo e socket de dados dos sensores, sair do programa, acessar o manual do usuário e controlar a execução do vídeo. Este módulo também é responsável pela visualização do stream de

vídeo, o que é feito com a biblioteca vlcj, mencionada na seção 2.6. A figura 20 mostra a interface em funcionamento.

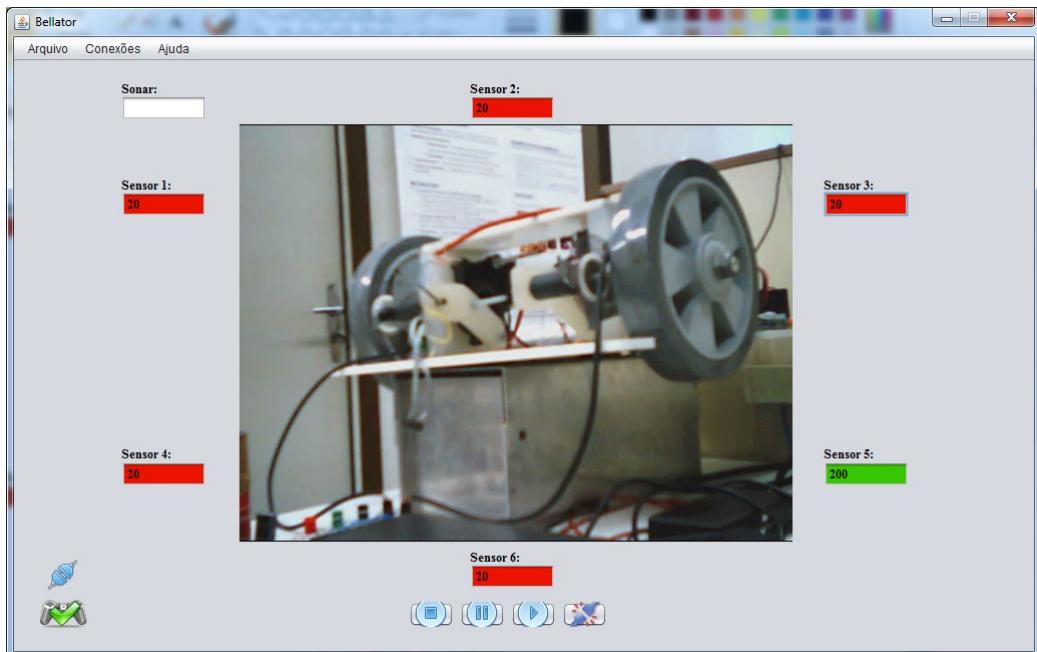


Figura 20: Interface do software Supervisor Remoto em fucionamento.

Para maiores detalhes sobre a implementação, consultar javadoc do software (disponível em (...)Bellator -> doc -> index.html). Para detalhes sobre a utilização do software, consultar o documento Manual do Usuário, o qual pode ser consultado no menu Ajuda -> Manual do Usuário através do próprio software.

4.3 Software de Baixo Nível

4.3.1 Migração para C8051F340DK

O primeiro passo planejado para o projeto, após a análise do estado do código em C para a placa 8051, foi realizar a migração do código para a nova plataforma de desenvolvimento, o C8051F340DK. Inicialmente, este processo parecia simples, porém a nova plataforma possui várias características que dificultaram a migração. Os principais problemas encontrados durante a migração foram:

- O clock diferente da nova plataforma, o que invalidou todas as configurações dos temporizadores, que são baseados no clock da placa e

controlam os outros dispositivos da placa.

- A mudança da nomenclatura para o acesso de diversos registradores.
- Falta de familiaridade da equipe com a nova plataforma.

O processo de migração iniciou-se com a transferência do código C existente para a nova plataforma de desenvolvimento. Obviamente, devido aos problemas acima descritos, o código necessitava de vários ajustes. A primeira parte reconfigurada foi a comunicação serial, essencial para realização de testes. Para isto, os valores para a geração do Baud Rate foram recalculados, para adequação ao novo clock, e alguns registradores modificados, já que a nova placa tem duas interfaces seriais disponíveis (UART0 e UART1), o que implica em mais registradores, com nomes diferentes. Apenas a UART0 foi utilizada.

Após o funcionamento da comunicação por porta serial, a seção do código que interpreta os comandos recebidos da camada de alto nível foi adaptada para utilizar o protocolo de comunicação.

Em seguida, o conversor AD foi configurado para realizar a varredura dos sensores de distância IR, conectados ao port 2 da placa C8051F340DK. Em posse dos dados de conversão, foi implementado o envio destas informações à camada de alto nível, onde as mensagens seguem a estrutura do protocolo, descrito na seção 2.5.

Finalmente, foi reconfigurada a geração de PWM para os motores do robô.

Em todas as etapas citadas foram realizados testes, para conferir se a reconfiguração de uma função não havia afetado o funcionamento de outras.

4.3.2 Funcionamento do Software

Inicialmente, o software de baixo nível realiza as configurações da comunicação serial, do conversor analógico/digital e da geração de PWM.

A configuração da comunicação serial é feita com o Timer1, responsável por gerar um baud rate de 115200 bps, foi configurada ainda para utilizar 8 bits e não utilizar bit de paridade.

O funcionamento do software pode ser dividido em três partes principais:

- Geração de PWM.
- Varredura dos sensores de distância.
- Interpretação de comandos recebidos do software supervisor.

A geração de PWM tem 4 saídas:

- Pino P1.1: contém a PWM responsável por fazer a roda direita girar para frente.
- Pino P1.3: contém a PWM responsável por fazer a roda direita girar para trás.
- Pino P1.5: contém a PWM responsável por fazer a roda esquerda girar para frente.
- Pino P1.7: contém a PWM responsável por fazer a roda esquerda girar para trás.

A frequência dos sinais de PWM gerados é de 200Hz, logo o período é de 5ms. A geração de PWM é controlada na interrupção do Timer0. Existem duas variáveis para controlar a geração de PWM para cada motor, uma delas controla a largura (por quanto tempo dos 5ms do período o sinal de PWM estará ativo, um valor inteiro de 0 a 15 inclusive) e a outra controla o sentido (bit 0 ou 1, que representa para qual pino o sinal de PWM será enviado).

A varredura dos sensores de distância é controlada pelo Timer3 e funciona da seguinte forma: a cada 50ms, é disparada uma varredura, onde cada sensor tem seu valor de tensão (análogo), que está na faixa de 0 a 2.6V, convertido para um valor digital entre 0 e 255, através do uso do conversor AD da placa C8051F340DK. Os sensores estão conectados da seguinte forma na port 2 da placa:

- Sensor 1: pino P2.0;
- Sensor 2: pino P2.1;
- Sensor 3: pino P2.2;
- Sensor 4: pino P2.3;
- Sensor 5: pino P2.5;
- Sensor 6: pino P2.6.

Após a conversão A/D da leitura de cada sensor, o valor convertido é armazenado em um vetor; em seguida, a próxima leitura a ser convertida é configurada para o pino do próximo sensor, a não ser que tenha ocorrido a conversão da leitura do sensor 6. Quando acaba a varredura (para os seis sensores de distância instalados na placa), uma flag que indica a disponibilidade

de novas conversões tem seu valor alterado para verdadeiro. Posteriormente, o laço principal verificará essa flag, e então os valores convertidos serão enviados através da interface serial, utilizando o protocolo, como explicado na seção 2.5.

A interpretação de comandos recebidos funciona como segue: continuamente, a interface serial está sendo monitorada através da interrupção serial, em busca de mensagens no formato especificado na seção 2.5. Quando um fim de comando é recebido na interrupção, a flag que representa a detecção de um novo comando é acionada. No laço principal do programa, quando esta flag é avaliada como verdadeira, ocorre a interpretação do comando. No código atual, os seguintes comandos são interpretados:

- RODA_LEFT.
- RODA_RIGHT.
- STOP.

Para maiores informações sobre estes comandos, rever a seção do protocolo: 2.5.

5 Considerações Finais

5.1 Resultados

Após a execução do projeto, a equipe obteve os seguintes resultados:

- A geração de PWMs e leitura dos sensores pela C8051F340DK funciona.
- Apenas 1 dos 6 sensores de distância infra-vermelho instalados no robô apresentou defeito.
- A placa de roteamento funciona.
- Os motores funcionam de acordo com os PWMs gerados pela C8051F340DK.
- O PC embarcado realiza corretamente o roteamento das mensagens entre a camada supervisória e a camada de baixo nível, e, também, produz o stream de vídeo da webcam.
- O software Supervisor Remoto comunica-se corretamente com o PC embarcado e atende aos requisitos funcionais e não funcionais.

Além disso, após a montagem do robô, a equipe realizou testes do sistema como um todo, ou seja, do funcionamento de todas as camadas apropriadamente, assim como descritas na seção 2. A equipe chegou a conclusão que o sistema funciona, porém, não de forma autônoma. Como mencionado na seção 5.3, a bateria do robô Bellator não foi capaz de alimentar todos os componentes plataforma robótica. A partir dos testes, porém, a equipe concluiu que todo o sistema deve funcionar de forma autônoma, se corrigido este problema.

5.2 Conclusão

A realização deste projeto provou ser uma valiosa fonte de aprendizado para a equipe, tanto no âmbito de gerência de projetos quanto no de desenvolvimento, devido à variedade de conhecimentos agregados no mesmo. A proporção do projeto, a importância crítica do fator de integração de todos os módulos do projeto, o fato de o mesmo já estar em andamento e já possuir código implementado com pouca ou nenhuma documentação foram fatores essenciais para torná-lo desafiador para a equipe. A maior dificuldade da equipe ao desenvolver o projeto foi a falta de documentação do que já havia sido realizado previamente. Desta forma, para que as próximas equipes não

passem pelas mesmas dificuldades, a presente equipe voltou boa parcela de seus esforços para o desenvolvimento de uma documentação detalhada sobre o projeto.

5.3 Trabalhos Futuros

O robô Bellator é um sistema complexo, que possui diversos componentes cujo desempenho pode ser aprimorado, além de alguns problemas críticos a serem resolvidos. Na montagem realizada nesse projeto, nem todos os componentes previamente disponíveis na plataforma foram utilizados, como os encoders ou o sonar, e portanto existem diversas possibilidades para trabalhos futuros. Algumas sugestões para tais trabalhos serão listadas a seguir:

- Realimentação dos encoders. O robô possui dois encoders cujas informações não são utilizadas. Essa informação é útil para realimentar a lógica de movimentação do robô, e compensar o movimento das rodas quando necessário.
- Sonar e servomotor. O robô possui também um sonar montado em uma base com um servomotor para realizar varreduras. Estes componentes podem ser integrados ao resto do sistema, permitindo ao usuário que controle a varredura remotamente através do joystick.
- Alimentação do robô. O robô é alimentado por uma única bateria de 12 Volts, o que gera dois problemas. Primeiro, o consumo de todos os componentes do robô, somados, é demasiado alto, o que limita muito a sua autonomia. Segundo, a parte lógica e a parte de potência do robô possuem todas a mesma alimentação, e um pico de corrente ocasionado pelos motores facilmente derruba a fonte de alimentação do pc embarcado, e poderia inclusive danificar os componentes. Essas duas partes precisam ser devidamente separadas e ter alimentação própria, além de se comunicar exclusivamente através de optoacopladores.
- Comunicação Wireless. Toda a comunicação do robô com a camada supervisória é feita via Wireless. Embora conveniente até o momento, esta tecnologia apresenta algumas limitações, principalmente em relação a alcance e consumo, e tendo como maior vantagem a taxa de transmissão de dados. Uma taxa alta é necessária para a transmissão de vídeo, mas uma solução mais adequada pode ser pesquisada.

Referências

- [1] SILICON LABORATORIES; C8051F340DK Datasheet, disponível em: <<http://datasheet.octopart.com/C8051F340DK-Silicon-Laboratories-datasheet-9512.pdf>>. Acesso em: 09/6/2010.
- [2] GENIUS; iLook 300 - Especificações. Disponível em: <<http://www.genius-europe.com/en/produktdetail.php?ID2=83&ID=31&ID3=479>>. Acesso em: 09/06/2010.
- [3] SHARP CORPORATION; GP2Y0A02F98YK Datasheet . Disponível em: <http://document.sharpsma.com/files/gp2y0a02yk_e.pdf>. Acesso em: 09/06/2010
- [4] VIA TECHNOLOGIES; EPIA M-Series Mini-ITX Mainboard. Disponível em: <http://www.via.com.tw/servlet/downloadSvl?id=81&download_file_id=3300>. Acesso em: 09/06/2010.
- [5] EDIMAX; EW-7128g PCI Wireless Lan PC Card Datasheet. Disponível em: <<http://www.edimax.com/images/Image/datasheet/Wireless/EW-7128g/EW-7128g-Datasheet-10202008.zip>>. Acesso em: 09/06/2010.
- [6] GNU; General Public License V2.0. Disponível em: <<http://www.gnu.org/licenses/gpl-2.0.html>>. Acesso em: 09/06/2010.
- [7] GNU; General Public License V3.0. Disponível em: <<http://www.gnu.org/licenses/gpl-3.0.html>>. Acesso em: 09/06/2010.
- [8] VideoLAN; VLC media player. Disponível em: <<http://www.videolan.org/vlc/>>. Acesso em: 09/06/2010.
- [9] VLCJ; Java Bindings for the VideoLAN Media Player. Disponível em: <<http://code.google.com/p/vlcj/>>. Acesso em: 09/06/2010.
- [10] JNA; Java Native Access. Disponível em: <<https://jna.dev.java.net/>>. Acesso em: 09/06/2010.
- [11] JAVA Input API. Disponível em: <<https://jinput.dev.java.net/>>. Acesso em: 09/06/2010.

- [12] Berkley Source Distribution License. Disponível em: <<http://www.linfo.org/bsdlicense.html>>. Acesso em: 09/06/2010.
- [13] GNU; Lesser General Public License. Disponível em: <<http://www.gnu.org/licenses/lgpl-3.0.html>>. Acesso em: 09/06/2010.
- [14] OBJECT MANAGEMENT GROUP; Unified Modeling Language. Disponível em:<<http://www.uml.org/>>. Acesso em: 09/06/2010.

Anexo A - Protocolo

```
*****  
* Projeto RoboMovel  
* Arquivo: protocolo.h  
* Descricao: Reune os mapeamentos de dispositivos a serem utilizados  
* na comunicacao serial com a interface de controle  
* Autor: Douglas Melchioretto  
*  
* Versao Atualizada: 02/06/2010  
*****  
  
// ATENCAO ! OS VALORES COMPREENDIDOS DE 0X00h A 0X1Fh  
// SAO RESERVADOS PADRAO RS-232.  
  
#define ON 0x0001  
//#define TRUE 0x0001  
#define OFF 0x0000  
//#define FALSE 0x0000  
  
#define ENCODER_DIREITO 0x0020  
#define ENCODER_ESQUERDO 0x0021  
  
// Se implementar leitura em funcao de sensor,  
// precisa concatenar o estado..  
// ON ou OFF ou a medida 1BYTE (0 a 255 CM)  
// Identificacao dos Sensores  
  
#define SENSOR_OPTICO_0 0x0022  
#define SENSOR_OPTICO_1 0x0023  
#define SENSOR_OPTICO_2 0x0024  
#define SENSOR_OPTICO_3 0x0025  
#define SENSOR_OPTICO_4 0x0026  
#define SENSOR_OPTICO_5 0x0027  
  
#define ENCODER_VALUE_SIZE 0x0004 // 4 bytes (32 bits)  
// Numero de Bytes (4) - 32 bits - O Robo vai ter que  
// andar 1385 km para estourar esse contador :D  
// Tendo em vista que o encoder eh de 2000 pulsos por  
// volta da roda que tem 64,5 cm de comprimento  
// Se colocar apenas 2 bytes, soh podemos medir 21 m
```

```

// e 3 bytes (nada usual) - 5,4 km
// Ainda podemos economizar esse bytes na transmissao ,
// transmitindo apenas a diferenca....
// Dessa maneira utilizando 1 byte como tamanho maximo,
// pode-se descrever um deslocamento de ate 8,256 cm
// Vale lembrar que caso sejam modificadas as
// caracteristicas mecanicas do sistema de propulsao
// (rodas) essa conta deve ser refeita.

// Implementacao dos estados dos sensores ON/OFF

#define SENSOR_ONOFF_1      0x0030
#define SENSOR_ONOFF_2      0x0031
#define SENSOR_ONOFF_3      0x0032
#define SENSOR_ONOFF_4      0x0034
#define SENSOR_ONOFF_5      0x0035
#define SENSOR_ONOFF_6      0x0036
#define SENSOR_ONOFF_7      0x0037
#define SENSOR_ONOFF_8      0x0038
#define SENSOR_ONOFF_9      0x0039
#define SENSOR_ONOFF_10     0x003A
#define SENSOR_ONOFF_11     0x003B
#define SENSOR_ONOFF_12     0x003C
#define SENSOR_ONOFF_13     0x003D
#define SENSOR_ONOFF_14     0x003E
#define SENSOR_ONOFF_15     0x003F

// IDENTIFICADOR DO ANGULO PAN (PWM) DO SONAR ( 1 Byte 0 - 255 )
#define SONAR_PAN           0x0050
    // IDENTIFICADOR DA DISTANCIA DO SONAR (2 Bytes)
#define SONAR_DISTANCIA    0x0051
    // IDENTIFICADOR DA BUSSOLA ELETRONICA
#define BUSSOLA             0x0052
    // IDENTIFICADOR DE INFORMACAO DO GPS
#define GPS                  0x53
    // VERSAO NMEA
#define NMEA                 183
    // TAXA DE TRANSMISSAO PADRAO NMEA-0183
#define NMEA_SPEED          4800
    // Mais informacoes uteis sobre NMEA-0183
    // http://www.kh-gps.de/nmea.faq

```

```

// Tags que identificam comandos a serem enviados ao ROBO

// Nivel das PWMS de Movimentacao ou passos
#define RODA_LEFT          0xA0
#define RODA_RIGHT         0xA1

// #define QQCOISA          0xA2
// #define QQCOISA2         0xA3

// Vai definir o Angulo Panoramico (Esquerda-Direita)
// 127 = CENTRO
#define CAMERA_PAN          0xA4
// Informacao Absoluta - Vai definir Angulo de
// Inclinacao (Acima-Abaixo) 127 = CENTRO
#define CAMERA_TILT         0xA5
// Reservado para possivel uso futuro...
#define CAMERA_ZOOM         0xA6

#define SONAR_PAN_SET       0xA7

// Parada De Emergencia !!!!
#define STOP                0xFF
// Caso a conexao tcp caia ou um evento watchdog
// ocorra ( timeout nos pacotes keep-alive
// do sinal de monitoramento ) parara imediatamente
// o robo !

#define SONAR_DISPARO        0x00F1

// Solicita um disparo do sonar de maneira
// assincrona, ou seja fora do modo de varredura.
#define DISPAROS_MEDIA      0x000A

// Ativa modo de varredura automatica e envio das
// informacoes dos encoders, do sonar (e do angulo
// de PAN do sonar, tendo em vista que no modo
// varredura ele se modifica automaticamente)
#define SONAR_VARREDURA     0x00F2

```

```
#define SONAR_MEDIDA      0x00F3

// Ordem para RESET GLOBAL
#define RESET              0x00F0
// Fim de comando
#define FIM_COMMANDO       0x00FE

// Mascara que define o bit de sentido da PWM
#define MASCARA_SENTIDO    0x0080
// Mascara que define os bits de valor da PWM
#define MASCARA_PWM         0x007F
```