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

# Lista de Figuras

# 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 este 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 plataform Bellator. The focus of the project undertaken by this team was, starting with the work previously done on the robotic plataform, 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 realizando 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 do robô Bellator, que possui uma placa C8051F340DK na qual estão ligados os sensores e motores, um PC embarcado rodando Linux no qual está ligada uma Webcam e um dispositivo de comunicação Wireless. Além disso o sistema também inclui um PC genérico, capaz de executar aplicativos Java e receber comunicação wireless, onde o software supervisor remoto será executado e onde o robô poderá ser controlado pelo joystick. Estes elementos serão melhores descritos na próxima seção. A seguir, será apresentado um diagrama do sistema e das interações entre os elementos do mesmo.

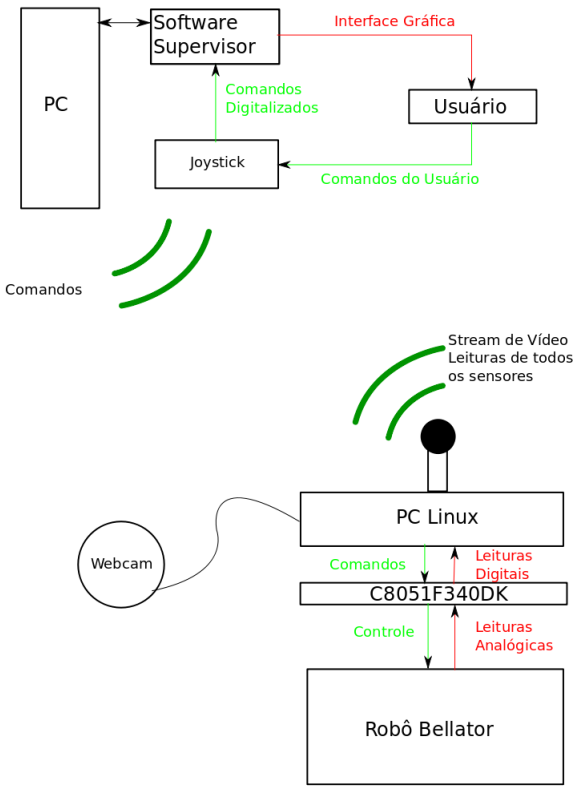


Figura 1: Diagrama Esquemático do sistema

A arquitetura do protótipo é dividida em três camadas lógicas denominadas baixo nível, alto nível e supervisória. A primeira é composta por seis

sensores ópticos, sonar, dois encoders de quadratura, dois motores DC e de uma placa C8051F340DK que faz o controle desta camada. Já a camada de alto nível é composta principalmente da plataforma PC embarcada, webcam e módulo wireless, e se comunica com a camada de baixo nível através de uma porta serial. Por último, a camada supervisória é composta por um PC que recebe pela rede sem fio as imagens e os dados dos sensores da camada de alto nível e as dispõe no monitor através do software supervisor remoto.

A partir do diagrama mostrado na figura 1, podemos obter uma visão geral do funcionamento do sistema. A placa C8051F340DK instalada no robô Bellator lê e converte leituras de tensão analógica vindas dos sensores, enviando valores digitais para o PC embarcado Linux através da conexão serial. Este, por sua vez, os envia utilizando um protocolo de comunicação através do dispositivo de comunicação wireless. Também via wireless, o stream de vídeo gerado pela webcam é enviado para o supervisor remoto. O software supervisor remoto, que é executado pelo PC externo, exibe as leituras recebidas e o stream de vídeo para o usuário através de sua interface gráfica. Com ajuda destas informações, o usuário pode tomar decisões sobre a navegação do robô, realizando as ações de movimento desejadas por meio de um joystick. O software supervisor recebe os comandos do usuário através do joystick e os envia, utilizando o protocolo de comunicação, através da comunicação wireless para o PC embarcado Linux. Este encaminha os comandos para a placa 8051 que, finalmente, traduz os comandos em sinais de controle para os dispositivos motores do robô Bellator.

## 2.2 Especificações

### 2.2.1 C8051F340DK

O C8051F340 é uma unidade micro controladora (MCU) equipada com um processador 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

- 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:

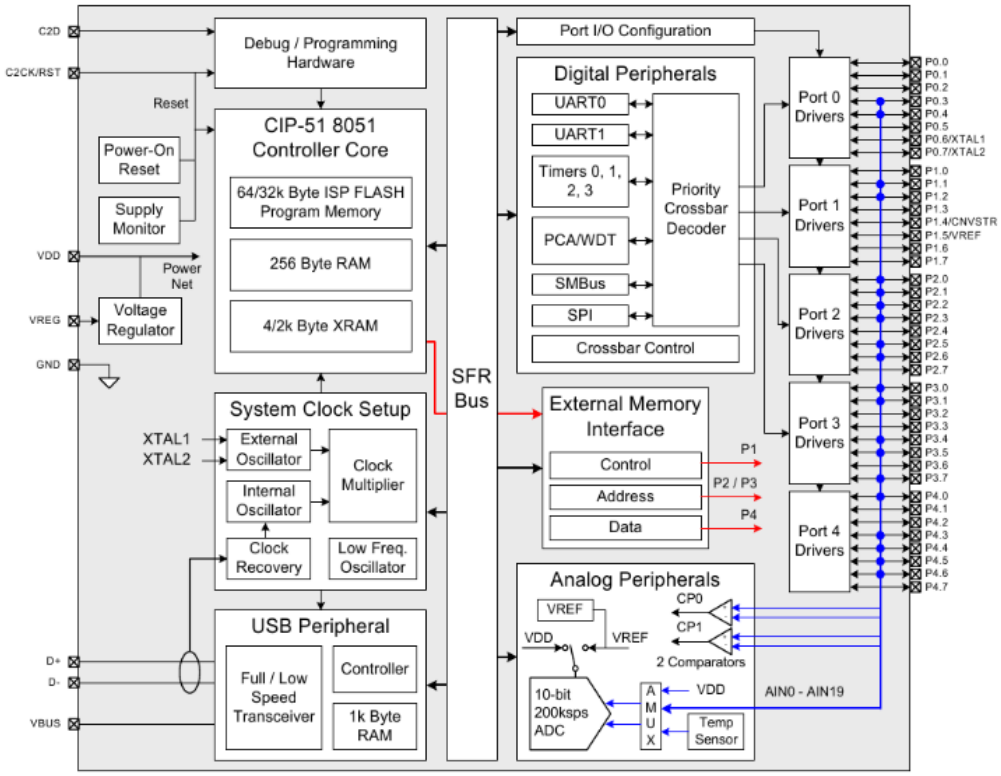


Figura 2: Diagrama em blocos do kit C8051F340DK. Fonte: datasheet.

### 2.2.2 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 pela 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 3.

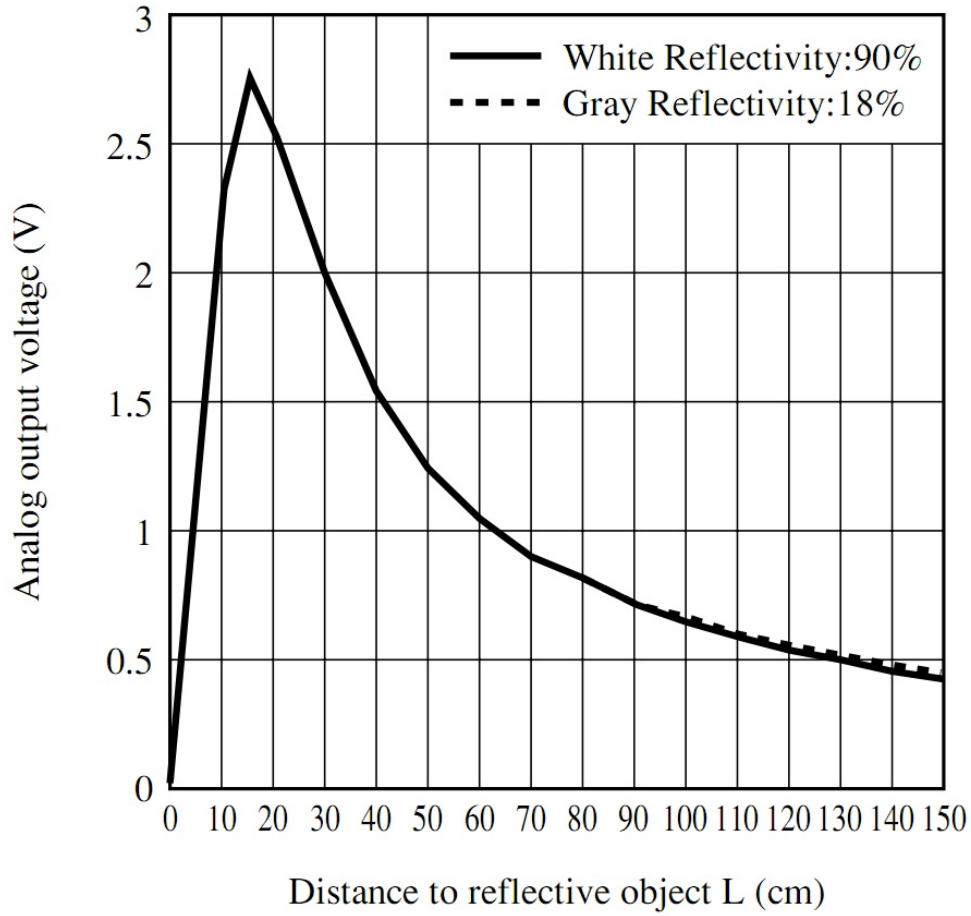


Figura 3: Curva de resposta do sensor de distância. Fonte: datasheet.

Pode-se observar que o modelo é pouco influenciado pelas cores dos objetos refletidos, isto devido ao método de medição baseado em triangulação [3]. 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 4.

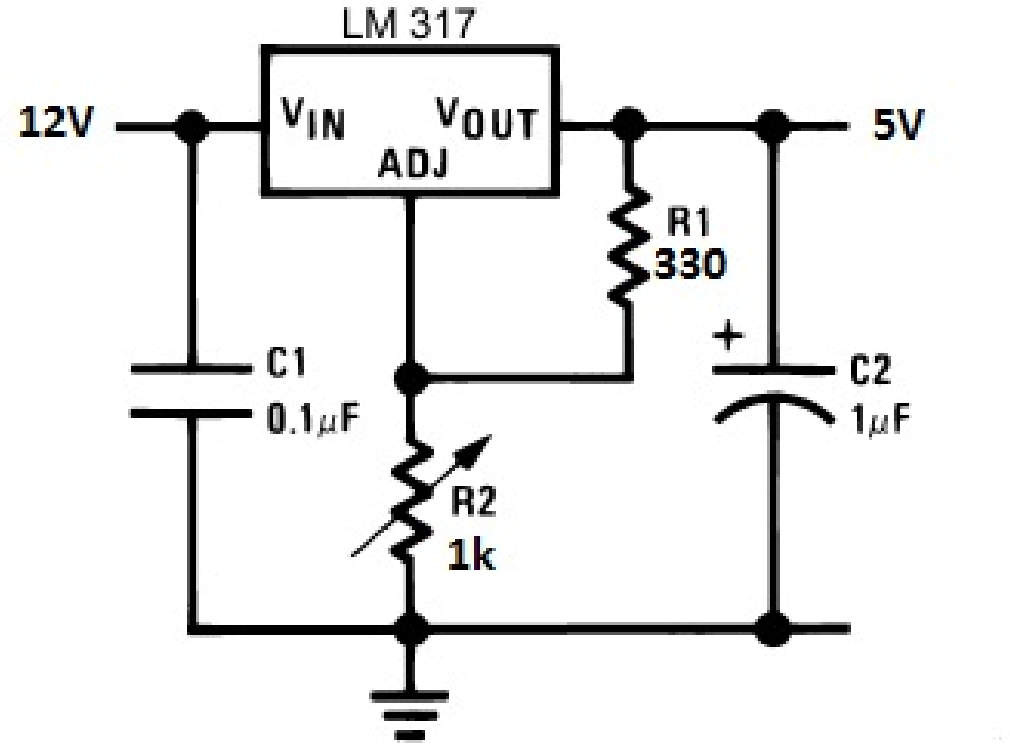


Figura 4: Diagrama esquemático do regulador de tensão dos sensores de distância.

As respectivas dimensões do sensor IR, em milímetros, são mostradas na figura 5

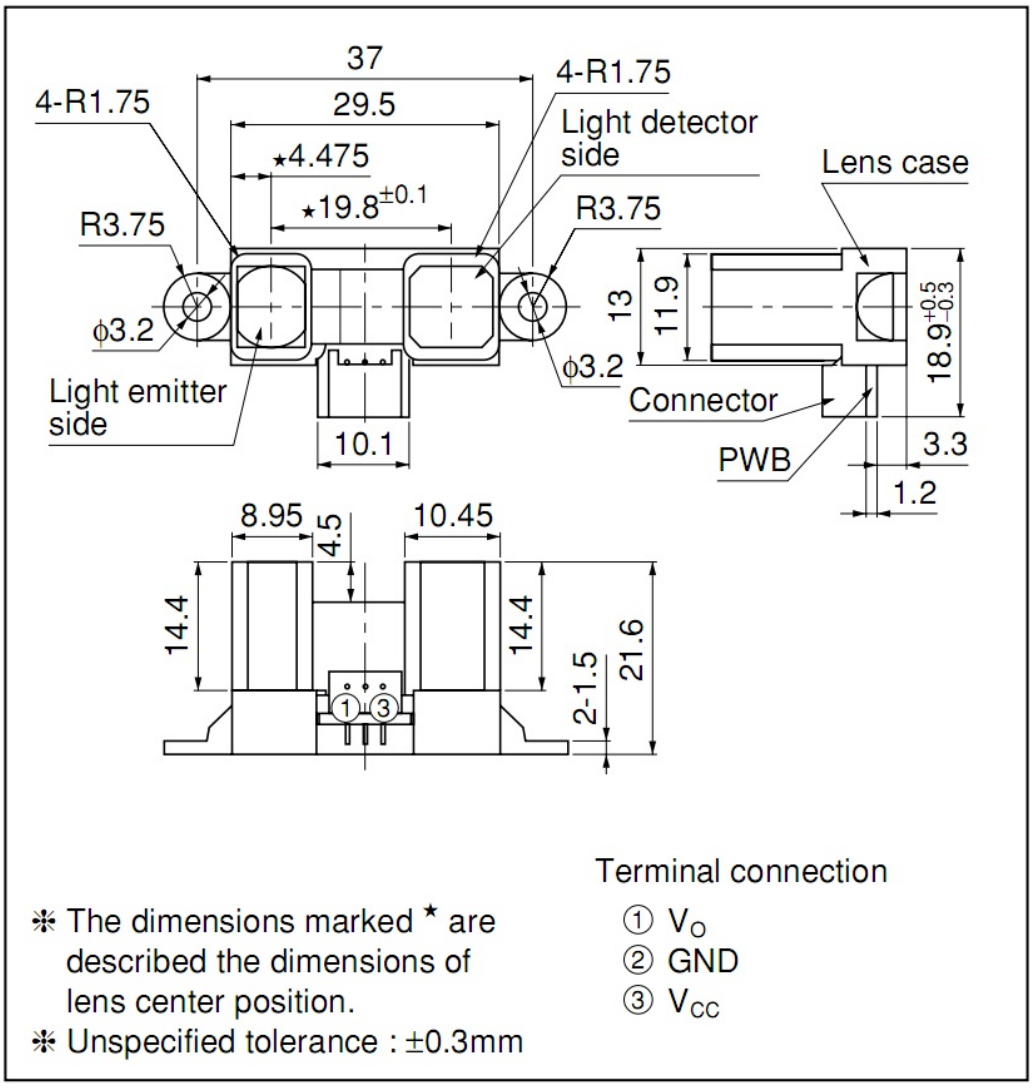


Figura 5: Dimensões do sensor GP2Y0A02F98YK. Fonte: datasheet.

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 menores distâncias.

### 2.2.3 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: 30, em uma resolução menor.

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.

### 2.2.4 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

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

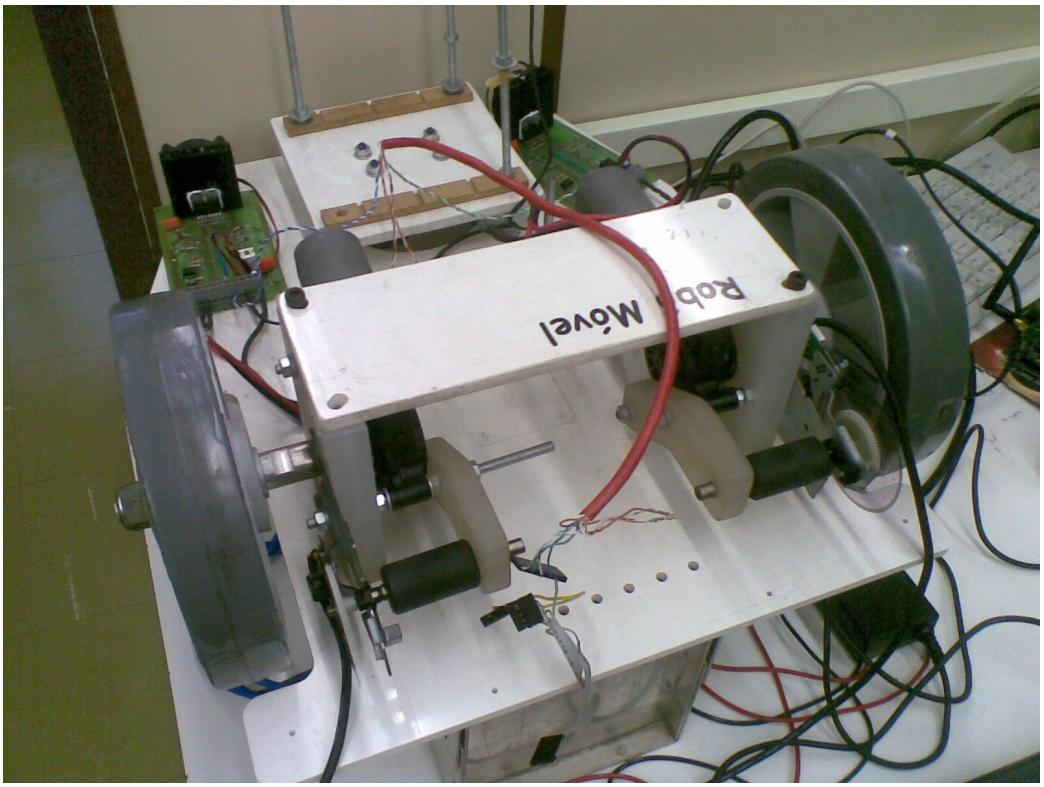

Figura 6: Foto do robô Bellator.

### 2.2.5 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:

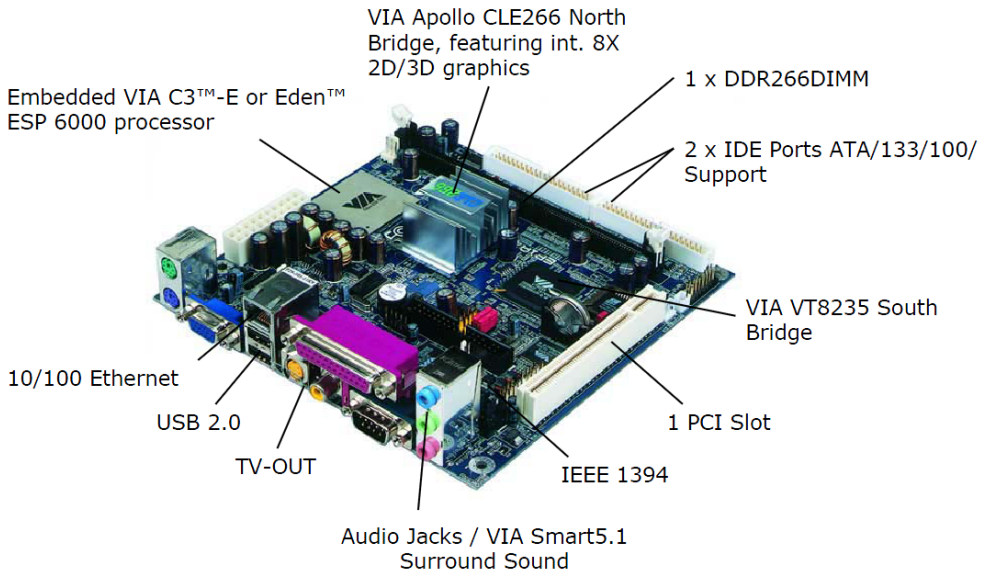


Figura 7: VIA EPIA M-Series MiniITX Mainboard. Fonte: Datasheet. pcembarcado

Mais informações a respeito da placa VIA EPIA M-Series Mini-ITX Mainboard podem ser encontradas no datasheet da mesma. [4]

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

### 2.2.6 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. Uma imagem do joystick pode ser visualizada na figura 8.


Figura 8: Joystick de Sony Playstation 2. Fonte <http://www.easytechnology.gr/images/PS2_sony_ps2_controller.jpg>

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, e se contiver componente no eixo X o movimento é de rotação (sentido horário para componente positiva, caso contrário sentido anti-horário).

## 2.3 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 joys-

tick, 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 desevolvimento 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.

### 2.3.1 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 do socket de dados.
- O software deverá possibilitar ao usuário a configuração da conexão do stream de vídeo.
- O software deverá possibilitar ao usuário o acesso à documentação do mesmo.

**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.

### 2.3.2 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 dispostos na figura 9

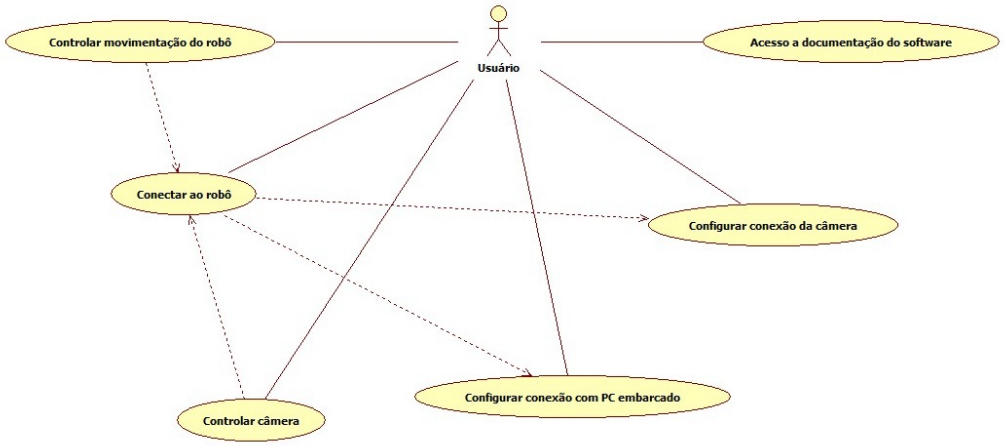


Figura 9: Diagrama de Casos de Uso.

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.

### 2.3.3 Diagrama de Classe

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 10. 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.

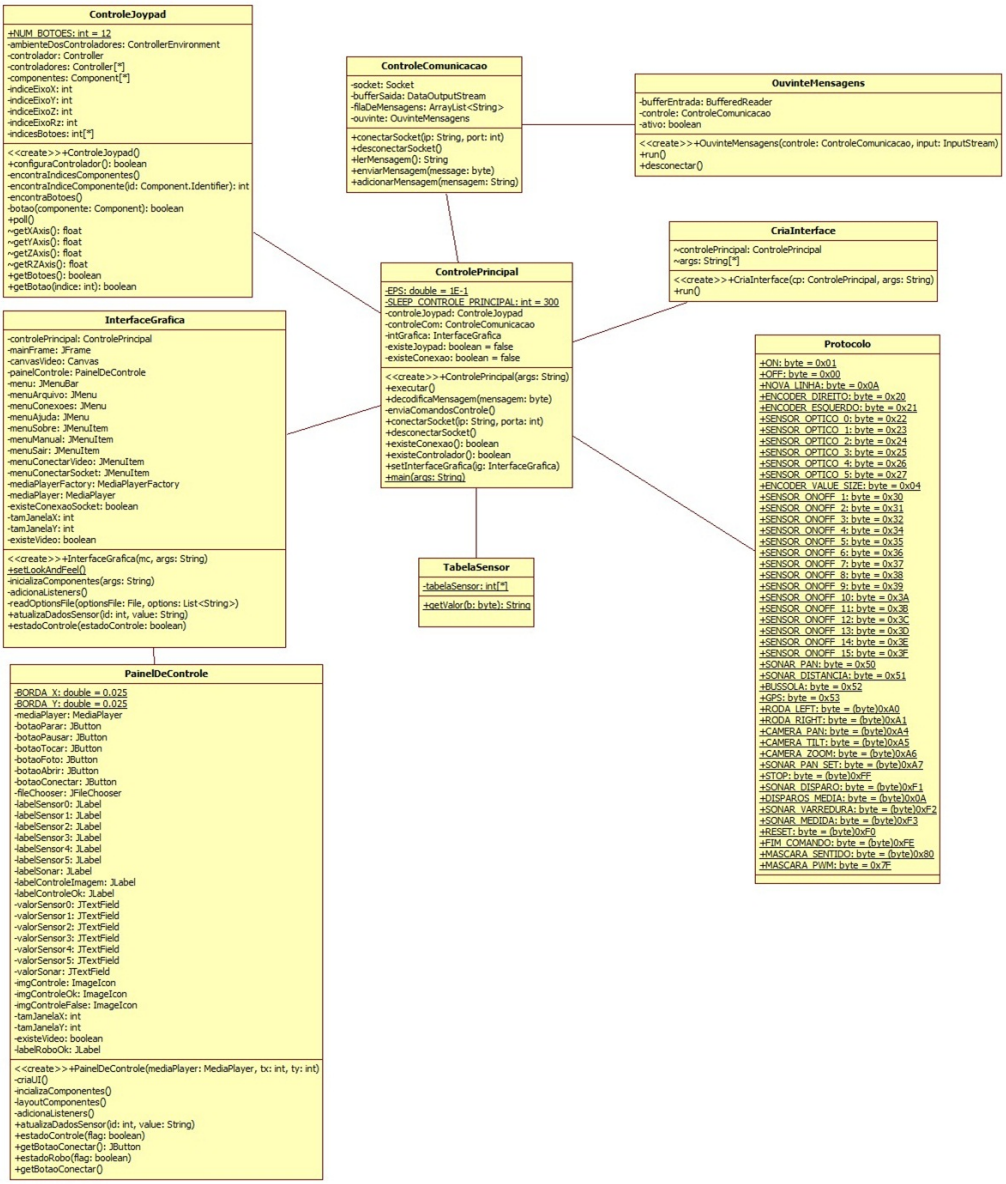


Figura 10: Diagrama de Classes.

Os detalhes de implementação do software serão abordados de maneira aprofundada na seção 3.3.

## 2.4 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, por sua vez, 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 gratituitamente sob a licença BSD [12].

# 3 Implementação do Projeto

## 3.1 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 89C51 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 89C51 para acionamento dos motores, controlável com comandos enviados através da porta serial.

- Sonar e servo-motor do sonar funcional, com leitura correta de distância.

- Recepção da imagem da Webcam pelo PC Linux, roteamento e acesso do stream por uma conexão Wireless está 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 2.3, visando um programa mais versátil e independente de plataforma.

## 3.2 Software de Baixo Nível

### 3.2.1 Migração para C8051F340DK

O primeiro passo planejado para a equipe, após a análise do estado do código em C para a placa 8051, foi migrar este 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. O código foi executado na nova plataforma sem alterações quanto à versão para a plataforma antiga, porém não funcionou. Um dos primeiros problemas encontrados pela equipe foi que a nova plataforma possuia Watchdog timer, um temporizador que, se não for desabilitado, reseta a placa periodicamente. Após a desativação deste timer, a equipe trabalhou no funcionamento da comunicação serial. Para isto, os timers para a geração do Baud Rate tiveram que ser recalculados, o que implicou na reestruturação da configuração da comunicação por porta serial. Devido a esta estruturação, a equipe decidiu por realizar a migração em partes, removendo temporariamente outras funções do código, tais como o sonar e servo-motor do mesmo.

Após o funcionamento da comunicação por porta serial, o código interpretador de comandos teve de ser adaptado para os padrões definidos no protocolo de comunicação desenvolvido pelo anterior responsável pelo projeto. Em seguida, foi implementada a leitura dos sensores ópticos de distância e envio dos dados dos mesmos de acordo com mesmo protocolo de comunicação. Neste ponto do desenvolvimento, a falta de familiaridade da equipe com a plataforma acarretou um atraso significativo no projeto, visto que fora suposto no cronograma que a migração não seria muito trabalhosa, notou-se que grande parte do software de baixo nível precisava ser reestruturada.

Com o funcionamento da comunicação serial, a equipe notou que a geração de PWM não estava mais operacional. A equipe, então, optou por re-implementar a rotina de geração de PWM, visto que, além disso, devido ao novo clock muito mais elevado que o da placa anterior, o PWM seria gerado com frequência incorreta.

### 3.2.2 Funcionamento do Software

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

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

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

A geração de PWM tem 4 saídas:

- Pino P1.4: contém a PWM responsável por fazer a roda direita girar para frente.
- Pino P1.5: contém a PWM responsável por fazer a roda direita girar para trás.
- Pino P1.6: 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) e a outra controla o sentido (qual pino terá o sinal de PWM gerado).

A interpretação dos dados recebidos dos sensores de distância é controlada pelo Timer3 e funciona da seguinte forma: a cada 50ms, ocorre a varredura dos sensores, onde cada um tem seu valor de tensão (analógico), que está na faixa de 0 a 2.6V, convertido para um valor digital entre 0 e 255, através do uso do conversor ADC da placa C8051F340DK. Os sensores estão conectados da seguinte forma na 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 definido, mandando mensagens da seguinte forma:

SENSOR_OPTICO_X VALOR_CONVERTIDO FIM_DE_COMANDO

A interpretação de comandos recebidos funciona como segue: continuamente, a interface serial está sendo monitorada através da interrupção serial, em busca de comandos. Um comando é composto por um byte indicando qual o tipo de comando, 0 ou mais bytes indicando um valor para o comando (caso necessário) e um byte de fim de comando. 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: este comando controla o sentido de rotação e valor de pwm para a roda esquerda.

- RODA_RIGHT: comando responsável por controlar sentido de rotação e valor de pwm da roda direita.

- STOP: faz o robô parar (zerando tanto o pwm da roda esquerda quanto o da roda direita).

## 3.3 Software Supervisor Remoto

O software supervisor remoto, como citado anteriormente, é responsável pelo interfaceamento do sistema como um todo com o usuário. A interface do programa pode ser observada na figura 11.

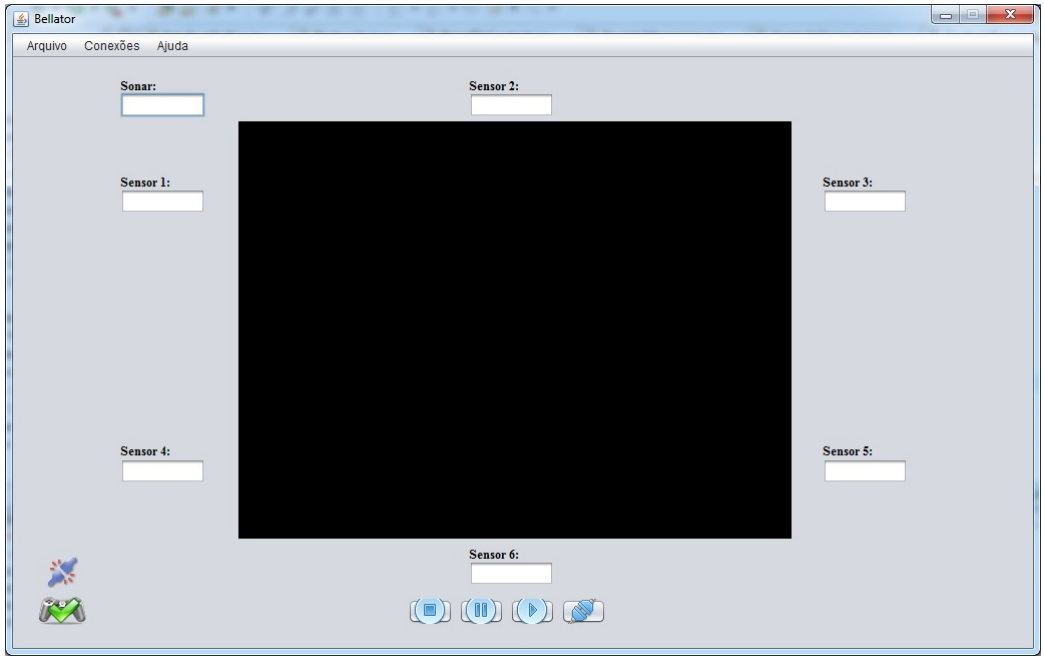


Figura 11: Interface do software Supervisor Remoto.

Este pode ser dividido em basicamente cinco núcleos funcionais: Principal, Controlador, Interface Gráfica, Comunicação e Util, as quais serão detalhadas nas seções a seguir.

### 3.3.1 Util

O núcleo Util consiste basicamente em dados a serem consultados pelo software durante sua execução, dados os quais correnspondem ao protocolo de comunicação utilizado a troca de mensagens entre o 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 é um vetor de bytes e deve respeitar este protocolo, que consiste na formatação da mesma para envio/recebimento da seguinte maneira:

- mensagem[byte 0]: Tipo de comando
- mensagem[byte 1]: Dados a ser enviado/recebido
- mensage[byte 2]: Fim de comando
- mensagem[byte 3]: Indicador de nova linha (\n)

Os bytes indicadores utilizados na mensagem são definidos no arquivo Protocolo.

### 3.3.2 Principal

O núcleo Principal é responsável pela coordenação da execução do software, ou seja, é este núcleo que faz a agregação de todos os núcleos para formar um só sistema. A execução do software inicia com a chamada do método executar da classe ControlePrincipal. Este consiste na thread que executa o laço principal do programa, o qual é responsável pela leitura, decodificação das mensagens recebidas pelo módulo de Comunicação; atualização dos dados na interface gráfica e; encaminhamento dos comandos do joystick, capturados através do módulo Controlador, para a camada de alto nível utilizando novamente o núcleo de Comunicação do software.

Para envio dos comandos realizados pelo usuário, primeiramente o software captura valores do joystick através de polling, os quais pertencem ao intervalo [-1, 1], 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, que no é baseada no protocolo 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;
- mensage[byte 2]: FIM_COMANDO;

- 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
- 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;

### 3.3.3 Controlador

O núcleo 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. Grande maioria dos métodos implementados neste módulo são baseados no uso da biblioteca JInput, citada na seção 2.4.

### 3.3.4 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 obter 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.

### 3.3.5 Interface Gráfica

O núcleo 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 2.4. A figura 12 mostra a interface em funcionamento.

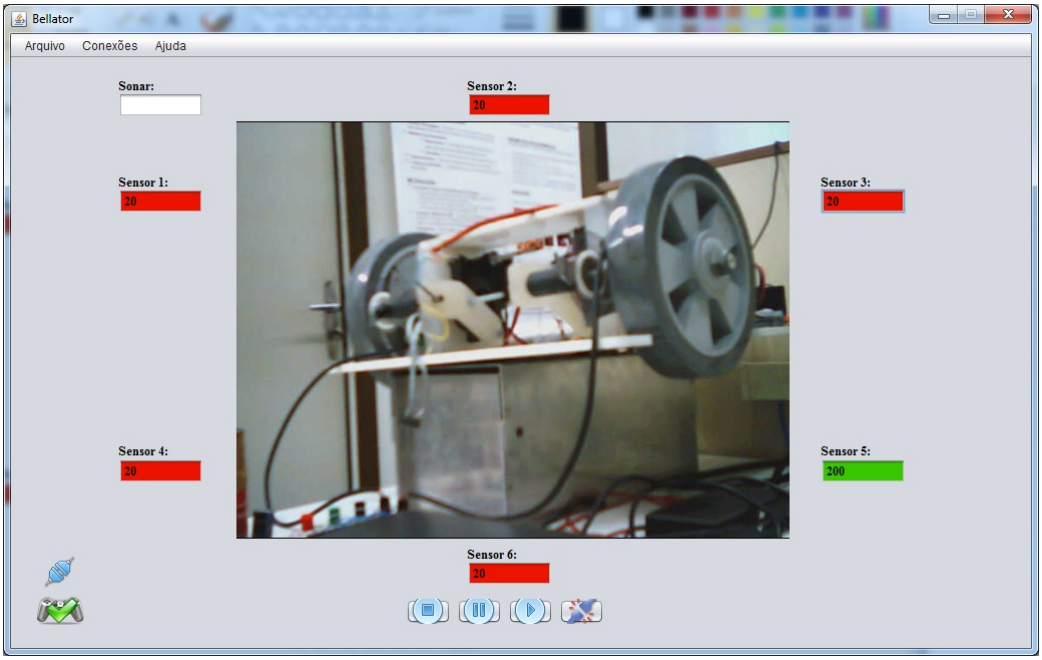


Figura 12: Interface do software Supervisor Remoto em fucionamento.

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.

# 4 Considerações Finais

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.

# 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.