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

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

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

SEGUNDO ENTREGÁVEL

CURITIBA

2013

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

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

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

CURITIBA

2013

SUMÁRIO

1 SEGUNDO ENTREGÁVEL - 27/03/2013 3
2 MODELAGEM UML 4
2.1 REQUISITOS FUNCIONAIS 4
2.2 REQUISITOS NÃO FUNCIONAIS 4
2.3 CASOS DE USO IDENTIFICADOS 5
2.4 DIAGRAMA DE CLASSES 7
2.5 DESCRIÇÃO DAS CLASSES 8
2.5.1 Pacote *visual* 8
2.5.2 Pacote *dados* 8
2.5.3 Pacote *comunicacao* 9
2.5.4 Pacote *gui* 10
2.6 FLUXO DE DADOS 10
2.7 DIAGRAMA DE ESTADOS 12
2.8 PROTOCOLO DE COMUNICAÇÃO 12
3 DIAGRAMA DE BLOCOS DO HARDWARE 16
3.1 DIAGRAMA DE BLOCOS 16
3.2 DIAGRAMA ELÉTRICO/ELETRÔNICO 17

1 SEGUNDO ENTREGÁVEL - 27/03/2013

Conforme estabelecido na relação de *deliverables* do documento de análise tecnológica, este segundo entregável consiste nos seguintes itens:

1. Versão inicial do diagrama de casos de uso (software embarcado).
2. Versão inicial do diagrama de fluxo de dados (software embarcado).
3. Versão inicial dos diagramas de estados (sistema de comunicação).
4. Versão inicial da descrição das mensagens e codificações dos comandos (sistema de comunicação).
5. Versão inicial do diagrama elétrico/eletrônico (hardware).

2 MODELAGEM UML

2.1 REQUISITOS FUNCIONAIS

1. A estação base deve mostrar na interface gráfica um mapa 2D (atualizado automaticamente) representando o robô e os obstáculos detectados por ele. Representado pelo requisito funcional: **"Estação base mostra mapa 2D do robô e dos obstáculos detectados – RF1"**.
2. O usuátio pode salvar o mapa 2D no disco rígido. Representado pelo requisito funcional: **"O usuário pode salvar o mapa – RF2"**.
3. O usuário pode carregar o mapa 2D do disco rígido. Representado pelo requisito funcional: **"O usuário pode carregar o mapa – RF3"**.
4. A estação base deve mostrar na interface gráfica a imagem captada pela *webcam* do robô. Representado pelo requisito funcional: **"Estação base mostra a imagem captada pela webcam – RF4"**.
5. O usuário pode movimentar o robô, controlando a velocidade de suas rodas remotamente pelo teclado da estação base. Representado pelo requisito funcional: **"O usuário pode movimentar o robô – RF5"**.
6. A estação base deve ser capaz de estabelecer conexão com o robô, informando o usuário caso a conexão ocorra com sucesso ou não. Representado pelo requisito funcional **"O usuário pode estabelecer a conexão entre o robô e a estação base – RF6"**.

2.2 REQUISITOS NÃO FUNCIONAIS

1. A imagem transmitida pela câmera do robô deve ser colorida. Representado pelo requisito não funcional: **"O robô deve enviar vídeo em imagem colorida para a estação base - RNF1"**.
2. O robô deve transmitir as imagens de sua câmera em tempo real. Representado pelo requisito não funcional: **"O robô deve transmitir os dados de vídeo captados pela**

câmera em tempo real - RNF2”.

2.3 CASOS DE USO IDENTIFICADOS

1. Visualização de mapa 2D na interface gráfica segundo os dados lidos dos sensores do robô. Representado pelo caso de uso: **“Mostrar mapa - UC1”**.
2. Gravação do mapa em um arquivo no disco rígido. Representado pelo caso de uso: **“Salvar mapa - UC2”**.
3. Leitura do mapa de um arquivo do disco rígido. Representado pelo caso de uso: **“Carregar mapa - UC3”**.
4. Leitura de informações dos sensores do robô. Representado pelo caso de uso: **“Ler amostras dos sensores - UC4”**.
5. Visualização de imagens da *webcam* do robô. Representado pelo caso de uso: **“Visualizar imagens da câmera - UC5”**.
6. Alteração pelo usuário da velocidade das rodas do robô. Representado pelo caso de uso: **“Alterar velocidade das rodas - UC6”**.
7. Solicitação de estabelecimento de conexão com o robô. **“Estabelecer conexão - UC7”**.
8. Consulta à documentação do robô pelo usuário. Representado pelo caso de uso: **“Consultar documentação do robô - UC8”**.

Foram produzidos três diagramas de casos de uso com base nesses casos de uso.

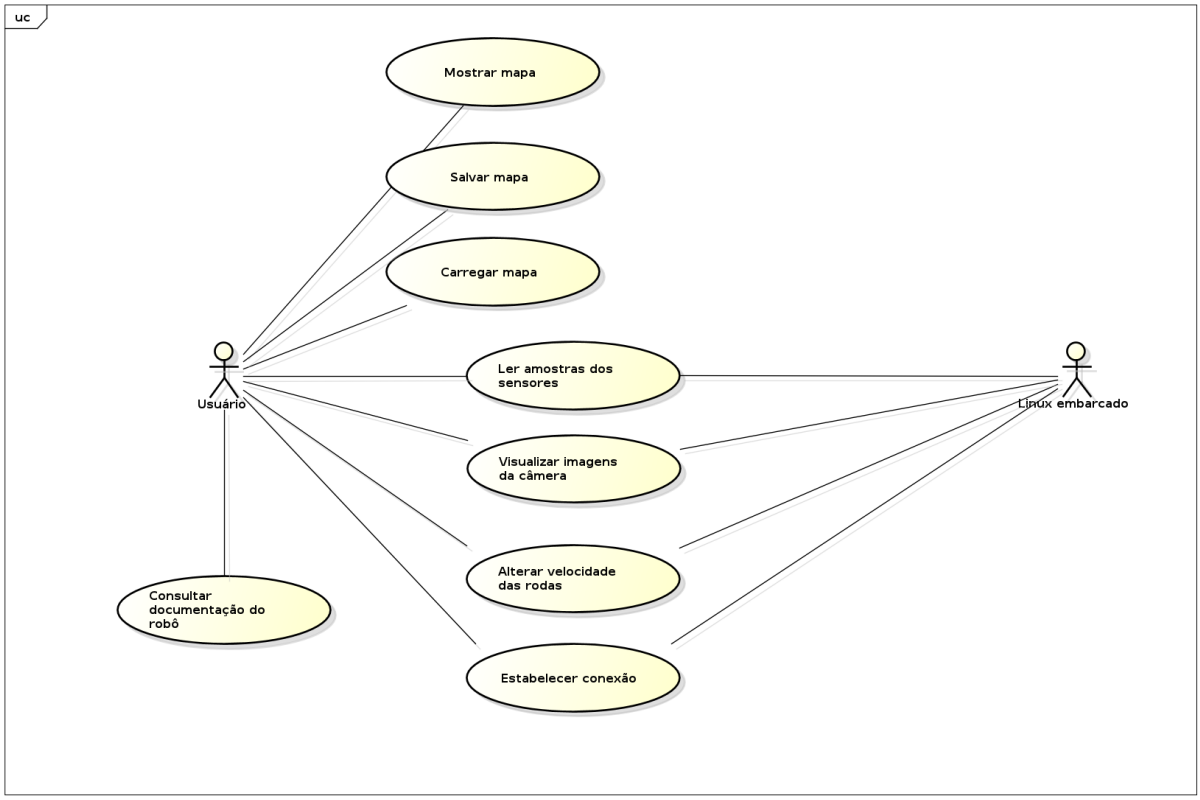

Figura 1: Diagrama de casos de uso do *software* da estação base.

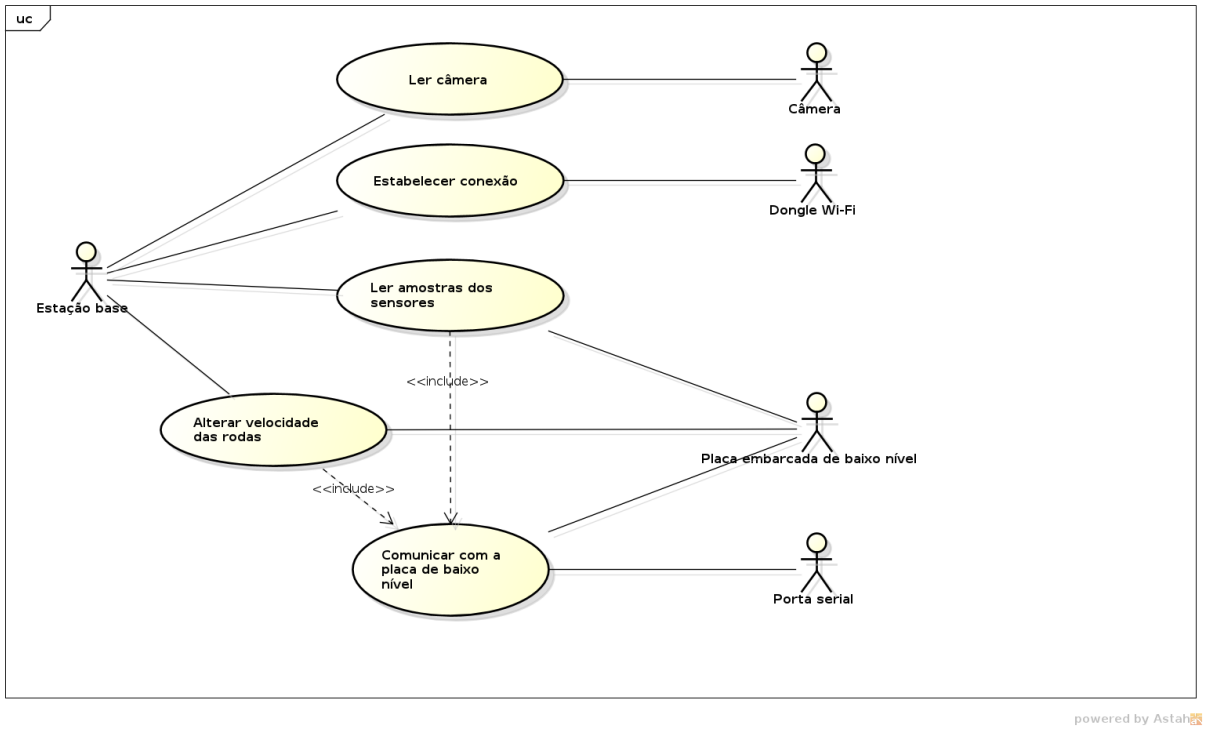

Figura 2: Diagrama de casos de uso do *software* para a placa TS-7260.

Figura 3: Diagrama de casos de uso do *software* para a placa com sistema embarcado.

2.4 DIAGRAMA DE CLASSES

A Figura 4 mostra o diagrama de classes da estação base.

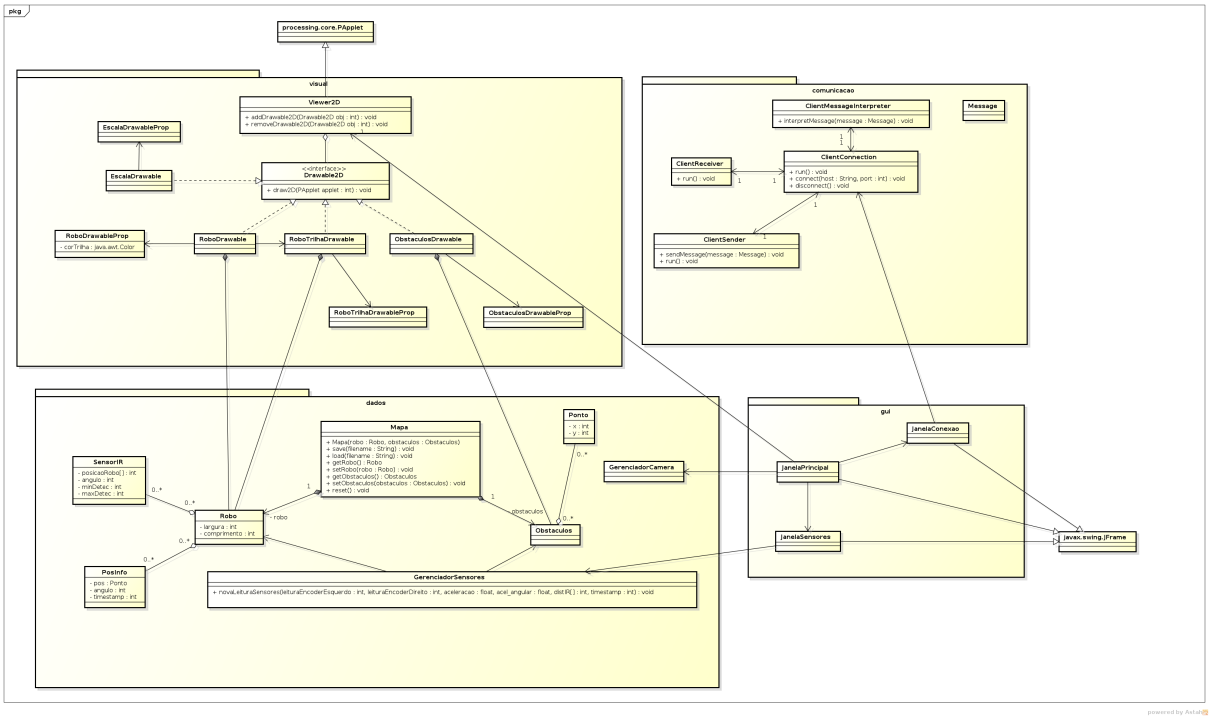

Figura 4: Diagrama de classes

2.5 DESCRIÇÃO DAS CLASSES

O software da estação base do robô foi dividido em cinco pacotes: *visual*, *dados*, *comunicao*, e *gui*. A seguir há uma descrição de cada pacote e das suas respectivas classes.

2.5.1 Pacote *visual*

Este pacote consiste de toda a parte visual da estação base e conta com as seguintes classes: Viewer2D, Drawable2D, EscalaDrawable, RoboDrawable, RoboTrilhaDrawable, ObstaculosDrawable, EscalaDrawableProp, RoboDrawableProp, RoboTrilhaDrawableProp e ObstaculosDrawableProp. Na Tabela 1 estão descritas as classes deste pacote.

2.5.2 Pacote *dados*

Este pacote consiste de toda a parte da estação base que processa e armazena as informações essenciais do robô e do mapa. Conta com as seguintes classes: Mapa, Obstaculos, Robo, ControleSensores, Posinfo, SensorIR e Ponto. Na Tabela 2 estão descritas as classes deste pacote.

Tabela 1: Pacote *visual*

Classe	**Descrição**
Viewer2D	Responsável por exibir os objetos Drawable2D. Possui recursos de pan, zoom e rotate.
Drawable2D	Representa genericamente objetos 2D que podem ser desenhados em um Viewer2D.
EscalaDrawable	Responsável por desenhar uma escala gráfica no mapa.
RoboDrawable	Responsável por desenhar o robô no mapa.
RoboTrilhaDrawable	Responsável por desenhar a trilha percorrida pelo robô no mapa.
ObstaculosDrawable	Responsável por desenhar os pontos de cada obstáculo no mapa.
EscalaDrawableProp	Contém as propriedades visuais de desenho da escala.
RoboDrawableProp	Contém as propriedades visuais de desenho do robô
RoboTrilhaDrawableProp	Contém as propriedades visuais de desenho da trilha do robô.
ObstaculosDrawableProp	Contém as propriedades visuais de desenho dos obstáculos.

2.5.3 Pacote *comunicacao*

Este pacote consiste em toda a parte de comunicação da estação base com o robô e conta com as seguintes classes: ClientMessageInterpreter ClientConnection, ClientReceiver, ClientSender e Message. Na Tabela 3 estão descritas as classes deste pacote.

É importante ressaltar que o protocolo TCP requer obrigatoriamente a especificação de um cliente e de um servidor para estabelecimento de uma conexão. Nas implementações desse protocolo em diversas linguagens (como Java e C++) existem tipos de *socket* distintos para cliente e servidor. Na criação de um *socket* de servidor, há obrigatoriamente a atribuição de uma porta de escuta, na qual o servidor aguarda que um cliente efetue uma requisição de conexão. Não é possível, ao menos nas implementações atuais do TCP, estabelecer conexão entre dois *sockets* de cliente ou entre dois *sockets* de servidor. Como neste projeto, o robô proverá serviços à estação base (envio de imagens da câmera, envio de leituras de sensores, além de prover a posibilidade de comando dos motores) o robô foi escolhido como servidor e a estação base como cliente. Enfatiza-se que o paradigma cliente-servidor não implica de forma alguma que a comunicação seja unidirecional. Pelo contrário, o envio de pacotes pode ser feito

Tabela 2: Pacote *dados*

Classe	**Descrição**
Mapa	Responsável por representar o mapa. Armazena as informações essenciais do robô e dos obstáculos detectados.
Obstaculos	Responsável por conter os obstáculos detectados pelo robô.
Robo	Responsável por representar o robô, este contêm largura, comprimento e centro de movimento (ponto central entre as duas rodas).
GerenciadorSensores	Responsável por atualizar a posição do robô e dos pontos que representam os obstáculos, de acordo com as leituras feitas pelos sensores.
Posinfo	Responsável por conter as informações de uma posição do robô.
SensorIR	Responsável por representar um sensor IR do robô.
Ponto	Representa um ponto de cordenadas cartesianas (x,y).
GerenciadorCamera	Responsável por gerenciar o status da câmera e o recebimento de imagens.

bidirecionalmente após uma conexão TCP ser estabelecida, sem nenhuma restrição quanto a isso.

2.5.4 Pacote *gui*

Este pacote consiste em toda a interface gráfica do sistema e conta com as seguintes classes: JanelaConexao, JanelaPrincipal e JanelaSensores. Na Tabela 4 estão descritas as classes deste pacote.

2.6 FLUXO DE DADOS

A Figura 5 mostra o diagrama de fluxo de dados.

Tabela 3: Pacote *comunicacao*

Classe	**Descrição**
ClientMessageInterpreter	Responsável pela interpretação dos comandos recebidos pelo cliente.
ClientConnection	Responsável por efetuar a gerência da conexão do cliente (estação base) com o servidor (robô).
ClientReceiver	Responsável por receber mensagens de um host de uma conexão.
ClientSender	Responsável por enviar mensagens ao host de uma conexão.
Message	Contém uma mensagem a ser enviada por um Sender.

Tabela 4: Pacote *gui*

Classe	**Descrição**
JanelaConexao	Janela com as informações e configurações da conexão com o Bellator.
JanelaPrincipal	Janela principal da interface gráfica da estação base.
JanelaSensores	Janela de configuração dos sensores.

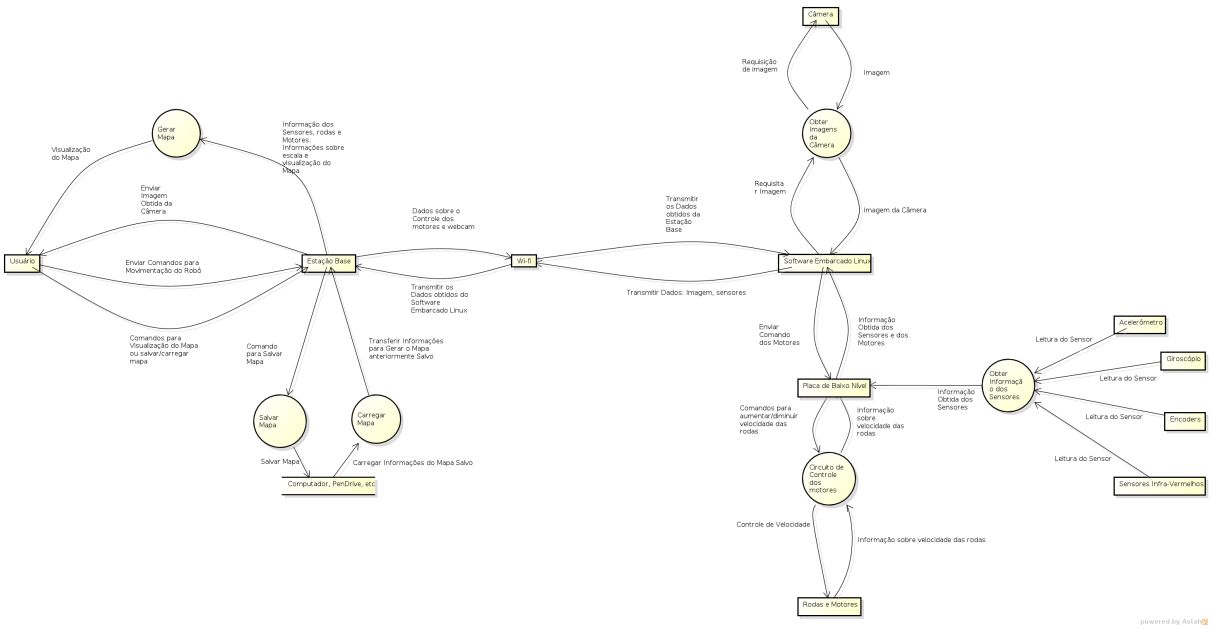

Figura 5: Diagrama de fluxo de dados.

2.7 DIAGRAMA DE ESTADOS

As Figuras 6 e mostram os diagramas de estados do sistema.

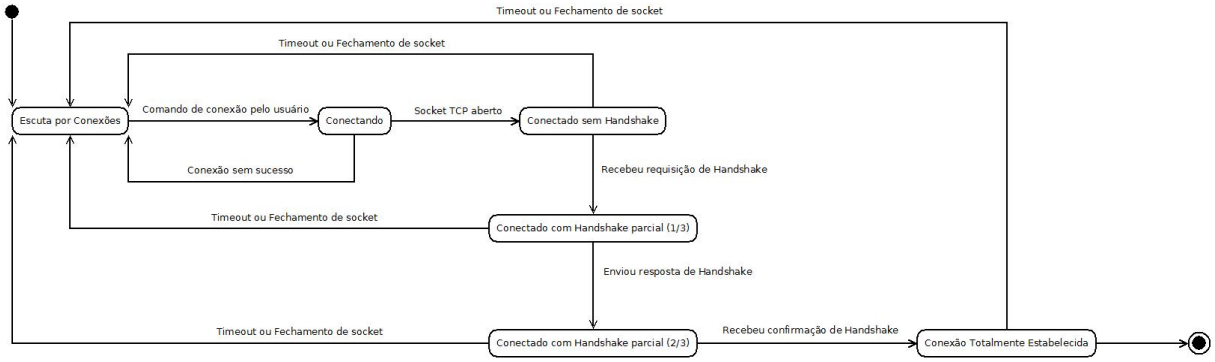

Figura 6: Diagrama de estados para a estação base.

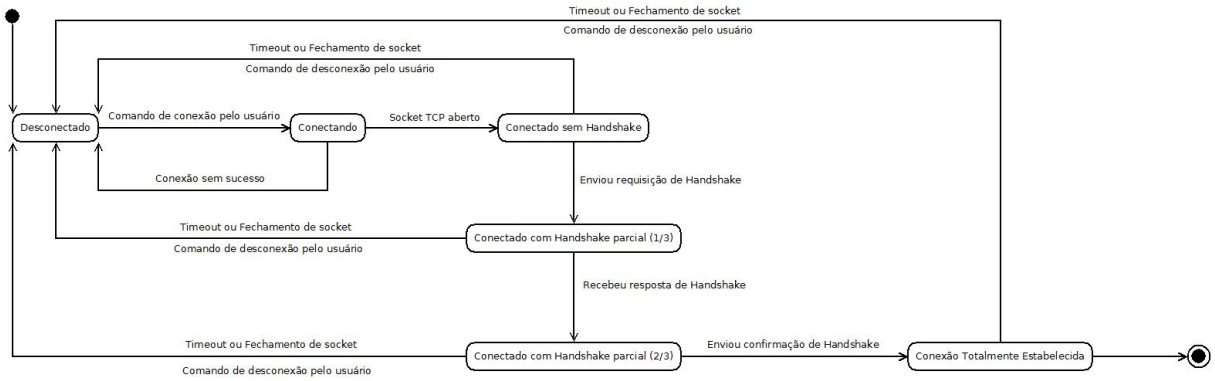

Figura 7: Diagrama de estados para o sistema embarcado.

2.8 PROTOCOLO DE COMUNICAÇÃO

Esta seção detalha o protocolo de comunicação estabelecido entre a estação base, a placa TS-7260 (sistema com linux embarcado) e a placa LPC2103 (sistema embarcado).

- Mensagens do TS-7260 para o LPC2103
  - **SYNC_END_CMD**: quando o microcontrolador LPC2103 recebe esta mensagem, responde com as leituras mais recentes de cada sensor de distância, em seguida as leituras a dos encoders;
  - **LEFT_WHEEL**: ao receber este comando, o microcontrolador utiliza o valor para definir o nível de PWM para a roda esquerda do robô valor e representado por apenas

um byte, onde o bit mais significativo indica o sentido de rotação da roda e os restantes a intensidade do PWM;

- **RIGHT_WHEEL**: funcionamento idêntico ao comando LEFT_WHEEL, e mas para a roda direita.
- **OPTICAL_SENSOR**:: representa a leitura de cada sensor, onde valor e um byte, cuja faixa de variação e [0, 255].
- **ENCODER**: representa a leitura de cada encoder, valor high e valor low juntos formam um inteiro de 16 bits que contém o valor da contagem do encoder.
- **IMU_DATA**: representa a leitura de cada sensor. O processo possui uma *timestamp*, para controlar a perda de dados. Os dados são lidos byte a byte. Os bytes que terminam com _H representam a leitura dos bits mais significativos. Aqueles que terminam em _L representam a leitura dos bits menos significativos. Aqueles que começam com A representam a leitura de um dos eixos do acelerômetro. Aqueles que começam com G representam a leitura de um dos eixos do giroscópio. A sequência ocorre da seguinte maneira: IMU_DATA; TIMESTAMP; AX_H; AX_L; AY_H; AY_L; AZ_H; AZ_L; GX_H; GX_L; GY_H; GY_L; GZ_H; GZ_L; END_CMD;

- Mensagens bidirecionais:
  - **ECHO_REQUEST**
    Requisição de ping.
  - **ECHO_REPLY**
    Resposta de ping.
  - **DISCONNECT**
    Solicitação de desconexão.

- Mensagens da estação base para o robô:
  - **HANDSHAKE_REQUEST**
    Solicitação de handshake.
  - **HANDSHAKE_CONFIRMATION**
    Confirmação de handshake.
  - **SENSORS_START**
    Solicitação de início da amostragem dos sensores.
  - **SENSORS_STOP**
    Solicitação de parada da amostragem dos sensores.

  - **SENSORS_RATE**

    (float) Nova taxa de amostragem

    Solicitação de mudança da taxa de amostragem dos sensores.

  - **SENSORS_STATUS_REQUEST**

    Resposta de status da amostragem dos sensores.

  - **WEBCAM_START**

    Solicitação de início da amostragem da webcam.

  - **WEBCAM_STOP**

    Solicitação de parada da amostragem da webcam.

  - **WEBCAM_RATE**

    (float) Nova taxa de quadros

    Solicitação de mudança da taxa de quadros da webcam.

  - **WEBCAM_RESOLUTION**

    (int) Largura em pixels

    (int) Altura em pixels

    Solicitação de mudança da resolução da webcam.

  - **WEBCAM_STATUS_REQUEST**

    Solicitação de status da amostragem da webcam.

  - **ENGINES_SPEED**

    (int) Nova velocidade da roda esquerda (Valor de 0 a 255)

    (int) Nova velocidade da roda direita (Valor de 0 a 255)

    Solicitação de mudança da velocidade dos motores.

  - **ENGINES_STATUS_REQUEST**

    Solicitação de status dos motores.

- Mensagens do robô para a estação base:

  - **HANDSHAKE_REPLY**

    Resposta de handshake.

  - **SENSORS_STATUS_REPLY**

    (boolean) Status da amostragem [on - off]

    (float) Taxa de amostragem

    Resposta de status da amostragem dos sensores.

– **WEBCAM_STATUS_REPLY**

(boolean) Nova taxa de amostragem

(float) Taxa de quadros

(int) Largura em pixels

(int) Altura em pixels

(boolean) Status da stream [on - off]

(int) Porta da stream

Resposta de status da amostragem da webcam.

– **ENGINES_STATUS_REPLY**

(int) Velocidade programada da roda esquerda (Valor de 0 a 255)

(int) Velocidade programada da roda direita (Valor de 0 a 255)

Resposta de status dos motores.

– **ENCODERS**

(int) Leitura roda esquerda

(int) Leitura roda direita

(long) Timestamp em milissegundos

Envio de leituras dos encoders.

– **ACEL_GYRO**

(float) Aceleração em X

(float) Aceleração em Y

(float) Aceleração em Z

(float) Aceleração angular em X

(float) Aceleração angular em Y

(float) Aceleração angular em Z

(long) Timestamp em milissegundos

Envio de leituras do acelerômetro e giroscópio.

– **OPTICAL_SENSORS**

(float[]) Distâncias detectadas pelos sensores ópticos

(long) Timestamp em milissegundos

Envio de leituras dos sensores ópticos.

3 DIAGRAMA DE BLOCOS DO HARDWARE

3.1 DIAGRAMA DE BLOCOS

Na figura 8 mostra-se o diagrama de blocos do sistema embarcado e suas conexões com o restante do robô. A seguir está também uma descrição para cada um dos blocos da placa de circuito impresso do sistema embarcado.

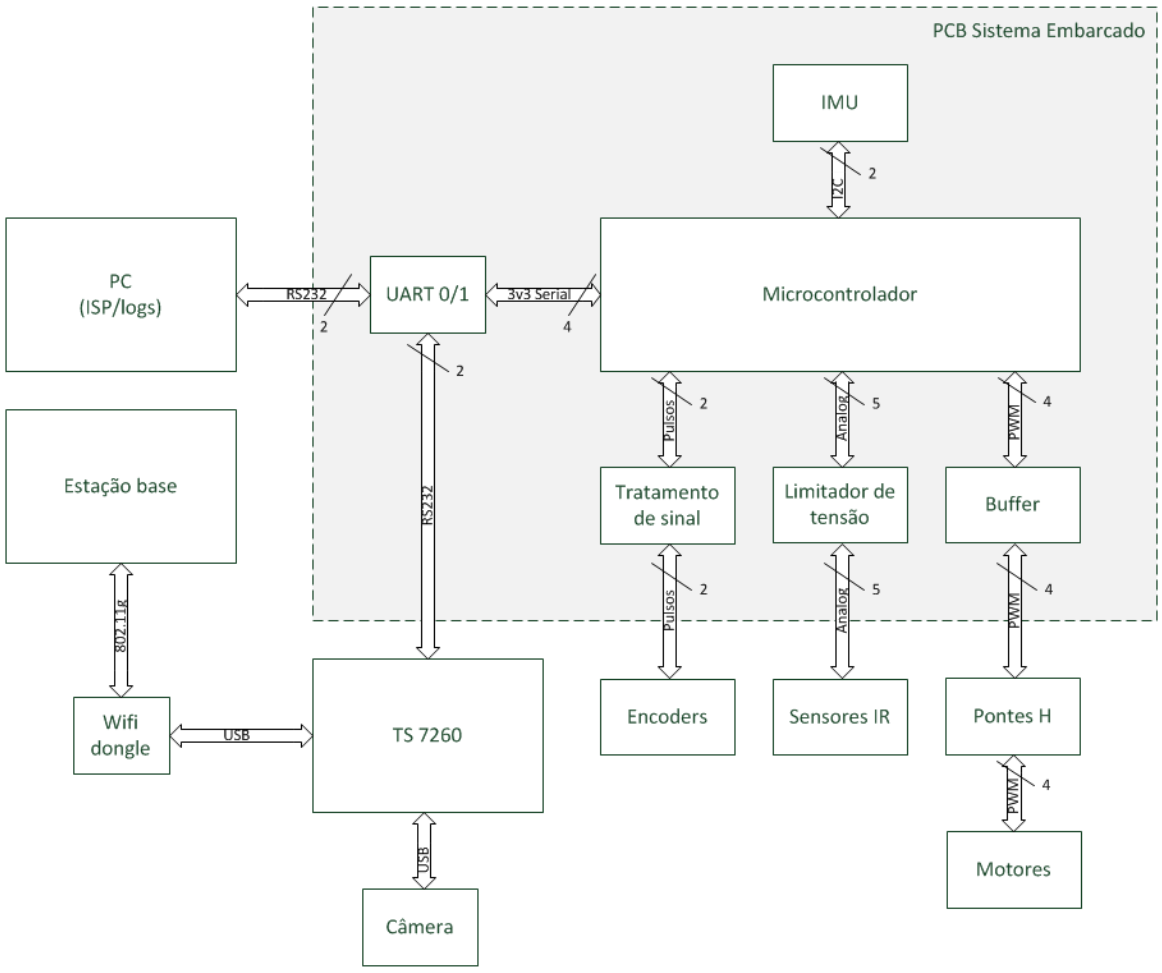

Figura 8: Diagrama de blocos do hardware

1. Microcontrolador: Este bloco fará a leitura dos sensores: encoders, infra-vermelhos, ace-

lerômetro e giroscópios. Além disso possui a implementação do protocolo de comunicação para interação com o linux embarcado da placa TS-7260.

2. UART 0/1: Responsável por ajustar os níveis de tensão para comunicação serial no padrão RS-232 com a placa TS-7260.
3. Buffer: Responsável por fornecer corrente e elevar os níveis de tensão de saída do microcontrolador de 3,3V para 5,0V. Esse buffer é conectado às pontes H já existentes no robô.
4. IMU: possui o acelerômetro e o giroscópio e se comunicará com o microcontrolador por meio do protocolo I2C.
5. Limitador de tensão: Necessário pois os sinais de saída dos sensores de infravermelho que já existem no robô não estão limitados em 5V, podendo a saída ultrapassar 5,0V e danificar o microcontrolador.
6. Tratamento de sinal: Composto por um filtro RC passa baixas e um schmitt trigger para remover qualquer falha que possa ocorrer na geração dos pulsos no encoder. A frequência de corte do filtro pode ser obtida pela velocidade máxima que o robô pode atingir, que foi suposta em 1 m/s (como apresentado nos requisitos de hardware).

3.2 DIAGRAMA ELÉTRICO/ELETRÔNICO

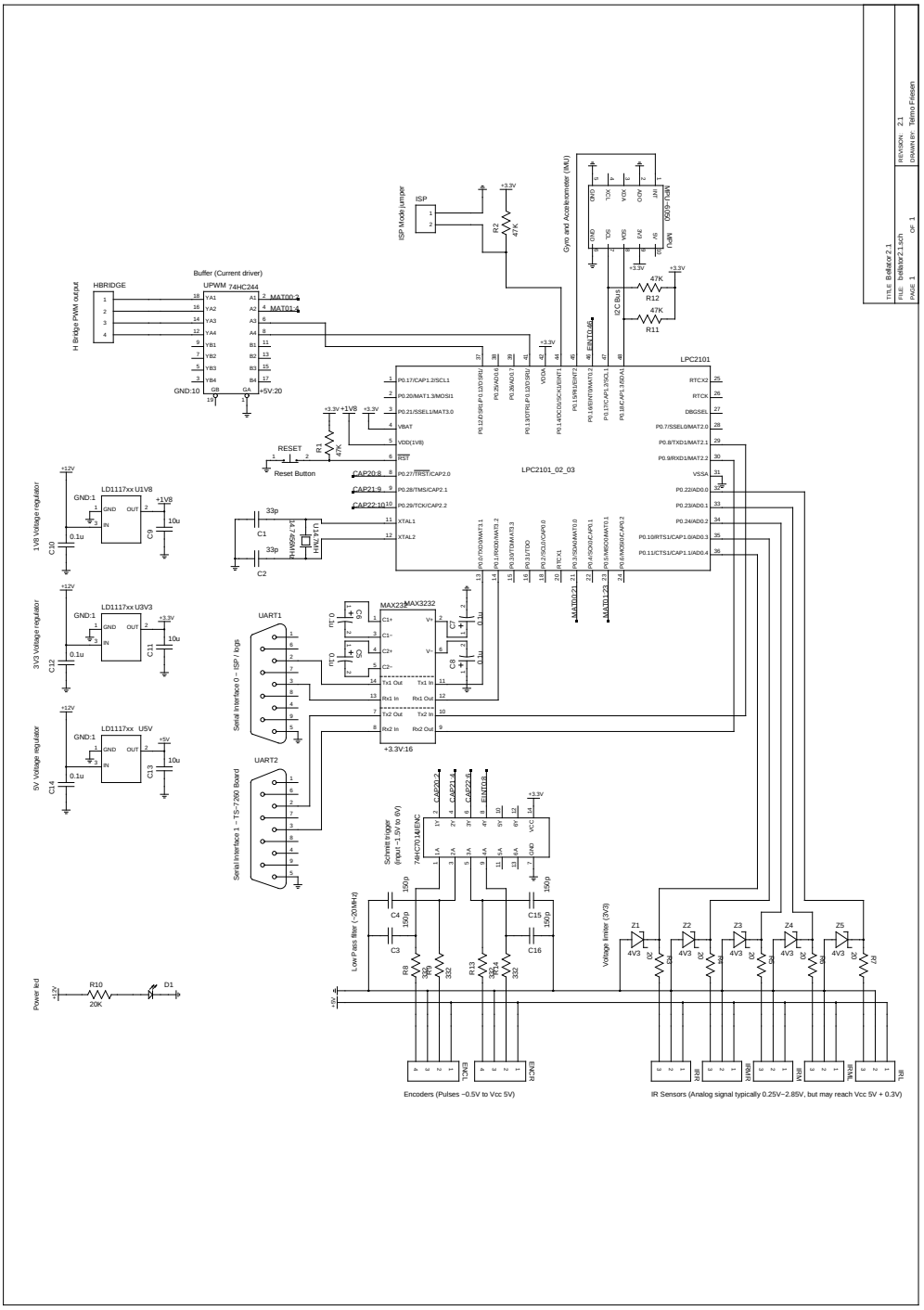

Figura 9: Diagrama de elétrico/eletrônico.