

Universidade Federal do Piauí – UFPI  
Campus Senador Helvidio Nunes de Barros - CSHNB  
Sistemas de Informação

**IMAGEBOT, UM APLICATIVO PARA RECEPÇÃO DE IMAGENS  
ENVIADAS POR ROBÔS MÓVEIS.**

Marco Antonio Marques Lima Filho

Picos-PI  
2013

Marco Antonio Marques Lima Filho

**IMAGEBOT, UM APLICATIVO PARA RECEPÇÃO DE IMAGENS  
ENVIADAS POR ROBÔS MÓVEIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí – UFPI, como requisito final para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Professor Mestre Algeir Prazeres Sampaio

Marco Antonio Marques Lima Filho

## **IMAGEBOT, UM APLICATIVO PARA RECEPÇÃO DE IMAGENS ENVIADAS POR ROBÔS MÓVEIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte para obtenção do Grau de Bacharel em Sistemas de Informação. Área de concentração: Sistemas de Informação.

Data de Aprovação: 11 / 04 / 2013.

Professor Mestre Algeir Prazeres Sampaio \_\_\_\_\_ UFPI  
Professor Mestre Rayner Gomes Sousa \_\_\_\_\_ UFPI  
Professor Especialista Ivenilton Alexandre de Sousa Moura \_\_\_\_\_ UFPI

Picos-PI  
2013

Eu, **Marco Antonio Marques Lima Filho**, abaixo identificado (a) como autor (a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI, 17 de abril de 2013.

FICHA CATALOGRÁFICA

Serviço de Processamento Técnico da Universidade Federal do Piauí  
Biblioteca José Albano de Macêdo

L732i Lima Filho, Marco Antonio Marques.  
ImageBot, um aplicativo para recepção de imagens enviada por robôs móveis / Marco Antonio Marques Lima Filho. – 2013.  
CD-ROM : il. ; 4 ¼ pol. (70p.)

Monografia(Bacharelado em Sistemas de Informação) Universidade Federal do Piauí. Picos-PI, 2013.  
Orientador(A): Prof. MSc. Algeir Prazeres Sampaio

1. Aforge.Net. 2. Aplicativo. 3. Recepção. I. Título.

CDD 005.1

Dedico este trabalho a meus pais Marco e  
Luiza, pessoas que me ensinaram a vencer na  
Vida.

## **AGRADECIMENTOS**

Agradecimento especial a minha mamãe Luisa Ferreira de Sousa, a meu pai Marco Antonio Marques Lima, meus irmãos Marciane Sousa e Marcelo Sousa Lima pelo apoio e incentivo aos meus estudos e a minha namorada Suzy Arianne por sua paciência e compreensão durante o desenvolvimento deste trabalho.

Aos meus colegas do curso de Sistema de Informação, especialmente Thiago José pelo apoio na realização dos testes necessários ao desenvolvimento do projeto.

A Marinete e Manoel Junior (In memória) por todo o apoio e ajuda quando eu cheguei à cidade de picos.

Aos professores que compõem o curso de sistemas de Informação, que contribuíram através de seus conhecimentos repassados.

Ao meu Orientador Mestre Algeir Prazeres Sampaio, obrigado por me ajudar a concluir esse trabalho.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original."

Albert Einstein

## RESUMO

Este projeto propõe o desenvolvimento de um aplicativo, para a recepção de imagens que serão enviadas por meio de uma câmera acoplada a um robô móvel, sendo sua transmissão realizada por tecnologia *wireless*. O aplicativo terá a capacidade de exibir imagens de ambientes que necessitem de vigilância e/ou monitoramento à distância em tempo real. Além de salvar as imagens em um banco de dados para análise dos ambientes sobre as condições insalubres ou perigosas. Na implementação do aplicativo (ImageBot), serão utilizados recursos do Visual Studio 2010 e *Sql Server* 2008 que servirão para desenvolver o algoritmo do aplicativo, bem como o *Framework Aforge.Net* utilizado na linguagem C# para o processamento da imagens que são capturadas por uma câmera. Estes são os requisitos básicos para alcançar o objetivo do projeto.

**Palavras-chave:** Aforge.Net, Aplicativo, Imagem, Recepção.



## **ABSTRACT**

This project proposes the development of an application to receive images that are sent through a camera attached to a mobile robot, and its transmission made by wireless technology. The application will have the ability to display images of environments that require monitoring and / or remote monitoring in real time. In addition to saving the image in a database for analysis environments on unhealthy or dangerous conditions. In the application deployment (ImageBot), will be used features of Visual Studio 2010 and Sql Server 2008 which served to develop the algorithm of the application as well as the Framework Aforge.Net used in C # for processing the images that are captured by a camera. These are the basic requirements to achieve the project objective.

**Keywords:** Aforge.Net, Application ,Image, Reception.

## LISTA DE FIGURAS

Figura 1 - Matriz de uma Imagem Digital .....	20
Figura 2 - Diagrama de Caso de Uso do Aplicativo (ImageBot) .....	23
Figura 3 - Robô Movél .....	24
Figura 4 - Imagem da Interface do Visual Studio 2010 .....	25
Figura 5 - Interface Principal do Aplicativo .....	29
Figura 6 - Imagem do Pannel Imgem do Ambiente .....	29
Figura 7 - Pannel de Conexão de Porta COM .....	29
Figura 8 - Pannel do Controle do Robô Movél .....	30
Figura 9 - Pannel do Controle da Câmera .....	30
Figura 10 - Pannel do Banco de Dados no Aplicativo .....	31
Figura 11 - Arquitetura de Comunicação .....	32
Figura 12 - Classe do Aforge.Video.DirecShow .....	33
Figura 13 - Trecho do Código de BuscaDispositivo .....	33
Figura 14 - Código para Captar Dados de Vídeos.....	34
Figura 15 - Trecho do Código Implementado para Clonar Quadro da Imagem.....	35
Figura 16 - Trecho do Código Para Fazer a Inserção da Imagem .....	35
Figura 17 - Trecho do Código para Converte Dados em Bytes.....	36
Figura 18 - Código Da Função Que Enviar Comando Pra Robô Móvel.....	39
Figura 19 - Arduino Duemilanove.....	37
Figura 20 - Módulos Xbee .....	38
Figura 21 –Câmera Sem Fio .....	39
Figura 22 - Placa EASYCAP e Receptor da Câmera sem Fio .....	39
Figura 23 - Pacote com os Arquivos Necessários para Execução do Aplicativo .....	40
Figura 24 - Imagem da Execução do Aplicativo na Captura da Imagem e o Armazenamento	42
Figura 25 - Robô Móvel Desenvolvido no LIPPO,Utlizado para os Testes.....	43

## **LISTA DE ABREVIATURAS E SIGLAS**

AUV	Autonomous Underwater Vehicle
USB	Universal Serial Bus
SQL	Structured Query Language
IDE	Integrated Development Environment
COM	Communication
LIPPO	Laboratório de Investigações e Pesquisas em Poéticas Digitais

# GLOSSÁRIO

**Arduino:** Uma plataforma de prototipagem eletrônica de hardware livre

**Byte:** Conjunto de 8 bits

**Pixel:** unidade de informação que descreve um ponto numa imagem Gráfica computadorizada.

**C##:** linguagem de programação

**Frame:** Quadro ou imagens fixa.

**Framework:** E conjunto que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

**Evento:** indicando que algo aconteceu, como por exemplo, o pressionamento de uma tecla.

**Shields:** são placas de circuito impresso normalmente fixados no topo do aparelho, através de uma conexão alimentada por pinos-conectores.

**Xbee:** Dispositivo de comunicação sem fio

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>14</b>
1.1 Motivação .....	15
1.2 Definições do Problema .....	15
1.3 Objetivos .....	16
1.4 Trabalhos Relacionados .....	16
1.5 Organização do documento .....	18
<b>2 FUNDAMENTAÇÃO TEÓRICA</b> .....	<b>19</b>
2.1 Robótica Móvel .....	19
2.2 Conceitos de Imagem Digital .....	20
2.3 Banco de Dados.....	21
2.3.1 Banco de dados SQL SERVER 2008 .....	21
<b>3 CONCEPÇÃO DO PROJETO</b> .....	<b>22</b>
3.1 Introdução .....	22
3.2 Análise dos Requisitos.....	22
3.3 Requisitos Funcionais .....	22
3.4 Caso de Uso .....	23
3.5 Ferramentas Utilizadas .....	24
3.5.1 Robô Móvel.....	24
3.5.2 Ambiente de Desenvolvimento .....	25
3.5.3 Aforge.Net Framework.....	26
3.5.4 Linguagem de Programação C# .....	26
<b>4 IMPLEMENTAÇÃO</b> .....	<b>28</b>
4.1 Introdução .....	28
4.2 Descrição do Software .....	28
4.2.1 Interface do Aplicativo.....	28
4.2.2 Arquitetura de Comunicação .....	32
4.2.3 Biblioteca Aforge.Video.DirectShow .....	32
4.2.4 Aquisição do Vídeo .....	33
4.2.4.1 Método BuscaDispositivo .....	33
4.2.4.2 Classe VídeoCaptureDevice.....	34
4.2.4.3 Classe para Clonar a Imagem do Vídeo.....	34
4.2.5 Banco de Dados BDImagem .....	35
4.2.5.1 Processamento da Imagem em Bytes.....	36
4.3 Descrição do Hardware.....	37
4.3.1 Arduino Duemilanove.....	37

4.3.2 Módulos Xbee.....	38
4.3.3 Câmera sem Fio .....	38
<b>5 ANÁLISE DOS RESULTADOS .....</b>	<b>40</b>
5.1 Instalação do Executável .....	40
5.2 Avaliação das Funções do Aplicativo.....	41
5.3 Ambiente de Teste .....	41
5.4 Execução do Aplicativo .....	42
5.5 Teste com Hardware.....	43
5.6 Teste do Banco de Dados.....	44
5.7 Dificuldades .....	44
<b>6 CONCLUSÃO E TRABALHOS FUTUROS .....</b>	<b>45</b>
<b>7 REFERÊNCIAS .....</b>	<b>47</b>
<b>8 APÊNDICE I.....</b>	<b>49</b>

# Capítulo 1

## Introdução

Nas últimas décadas, a utilização de sistema de exploração e monitoramento baseados em robô móvel tem sido objeto de muitas pesquisas devido a sua utilidade em diversas aplicações.

Atualmente com o avanço da tecnologia, os robôs móveis são utilizados em monitoramento, vigilância e na exploração de ambientes de difíceis acesso e remotos, entre outros. Desta maneira, utilizam sistemas de monitoramentos onde câmeras capturam imagens do ambiente. Um sistema de computador recebe as imagens que são transmitidas através de tecnologia *wireless*. A tecnologia *wireless* baseia-se em protocolos de comunicação de dados sem fios. Esta tecnologia tem como meio de transmissão a propagação de ondas eletromagnéticas no ar, utilizando a transmissão via radiofrequência para que a comunicação seja estabelecida.

O emprego de sistema de monitoramento é utilizado através de câmera de vídeo que captura a imagem do ambiente através da tecnologia *wireless* que permiti maior mobilidade de acompanhamento em tempo real de todas as situações respectivas ao ambiente visualizado.

É comum hoje utilizar sistema de monitoramento a distancia, utilizando câmera de vídeo sem fio. Neste contexto, o dispositivo *Xbee* (dispositivo de comunicação via radiofrequência), tem sido utilizado com o propósito de possibilitar o controle remoto de automação residencial e na exploração de ambientes.

O Processamento de imagens é um método de análise de dados multidimensionais, que manipula imagens para conseguir informações que possam conduzir á tomada de decisão, utilizando para reconhecimento de ambientes com aplicação na robótica para monitoramento e exploração de ambientes. Segundo (SBEGHEN, 2007), “O Processamento Digital de Imagens tem crescido muito nas últimas décadas, com a utilização de imagens e gráficos em uma grande variedade de aplicações, possibilitando a utilização de sistemas mais eficientes e mais baratos. Muitas áreas vêm utilizando Sistemas de Processamento Digital de Imagens como, por exemplo: reconhecimento de padrões, medicina, meteorologia, pesquisas espaciais etc.”.

Este trabalho trata-se do desenvolvimento de um aplicativo de recepção de imagem

enviada de uma câmera sem fio. O aplicativo permitira operar o robô ao mesmo tempo em que exibe imagens do ambiente e estas por sua vez fornecerão informações ao aplicativo.

## 1.1 Motivação

A Área da robótica móvel é utilizada de varias formas no cotidiano dos seres humanos. A criação de robôs autônomos tem possibilitado a execução de inúmeras tarefas como o trabalho de monitoramento dentro de túneis de esgotos, além de outras ações consideradas perigosas. O uso de técnicas de controle dos robôs móveis permite o agir em ambientes extremos e a tomada de decisões de forma autônoma. Além disso, esses dispositivos terão a capacidade de transmissão e recepção de informações sem a utilização de fios, por dispor de uma série de tecnologias de comunicação wireless.

Neste contexto a câmera acoplada a um robô móvel tem a importante função de controle e monitoramento do ambiente onde esta inserida, as imagens enviadas pela câmera servem para análise desse ambiente, além de permitir o controle do robô móvel, possibilitando assim o agir da forma correta e objetiva na exploração em situações de risco.

Neste trabalho será desenvolvido um aplicativo para recepção de imagem que executado em um computador auxilia nas tarefas de monitoramento de ambientes através de uma câmera onde as imagens são capturadas, permitindo uma melhor navegação e análise do ambiente.

## 1.2 Definições do Problema

A razão para a realização desta monografia foi á necessidade de recepção de imagens em tempo real enviadas por câmera acoplada a robôs móveis quando da execução de tarefas em ambientes remotos, para uma análise das imagens do ambiente. Assim há a necessidade da utilização de um aplicativo que funcione como central de recepção de imagens e que possam ser acessadas de um computador que funcione como base de controle e de armazenamento destas imagens em um banco de dados, localizado em ponto distante do dispositivo robótico através de comunicação wireless.

A monografia contribui para estudos futuros de trabalhos voltados a área da robótica com a utilização da tecnologia *wireless* e *Aforge.Net* para fazer o processamento de imagens do ambiente que esta sendo explorado. Tendo ainda importância na área da comunicação de sinais de imagens estáticas e dinâmicas, (fotos e vídeos) para os mais diversos fins.



### 1.3 Objetivos

O projeto tem por objetivo desenvolver um ambiente de recepção para imagens transmitidas por câmera através da utilização de tecnologia wireless em um robô aéreo ou terrestre para explorar e/ou monitorar ambientes insalubres que ofereçam riscos a seres humanos, permitindo assim, que as imagens possam ser exibidas na tela de um computador para serem salvas em um banco de dados. Além de ter o controle do robô e da câmera.

A monográfica busca a implementação de técnicas de processamento de imagem e de banco de dados. Para chegar ao objetivo geral foram necessários os itens abaixo mencionados:

- Conhecer os métodos de transmissão de imagem;
- Programar utilizando as linguagens C/C#;
- Programar na linguagem SQL serve 2008 para inserção, excluir a imagem no banco de dado do aplicativo;
- Conhecer métodos sobre Framework Aforge.Net;
- Simular aplicação no recebimento das imagens;
- Avaliar os resultados do software na captura das imagens.

### 1.4 Trabalhos Relacionados

CAVALCANTE, (2011) Aborda uma proposta de um sistema de monitoramento com o uso de câmeras de vídeo de baixo custo, as webcams. Estes dispositivos podem ser encontrados a preços bem acessíveis e estão frequentemente disponíveis nos computadores usados em ambientes doméstico e profissional. A disponibilidade, o baixo custo do equipamento e o acesso à internet banda larga, cada vez mais presente nos lares e empresas, garante o baixo custo da solução proposta. Acessível à maioria das pessoas, esse sistema oferece recursos de visualização em tempo real das suas imagens em qualquer local do mundo, utilizando navegadores *Web*.

Neste trabalho os resultados obtidos na implementação de um sistema de vigilância com câmeras de vídeo que tivesse um baixo custo, foi alcançado. Mostrou-se que, com apenas uma câmera web e um computador com internet banda larga é possível construir um sistema de monitoramento de câmeras em tempo real através da internet.

CESAR, (2002) Apresenta o desenvolvimento de um veículo subaquático autônomo

(AUV) que seja capaz de locomover-se de forma autônoma dentro de um tanque. O veículo deve possuir câmera embarcada, o que lhe permitirá desenvolver técnicas de processamento de imagem visando exploração e mapeamento do interior do tanque de água, de forma a fazer a detecção de objetos pela cor e forma. Onde tenta estabelecer um processamento de imagem para exploração do ambiente. Além da operação remota do veículo utilizando-se comunicação *wireless* que possibilita o acesso a informações sobre o veículo e o envio de comandos através de um computador.

O trabalho utilizou um sistema de captura e processamento de imagens e a programação embarcada no microprocessador para o controle remoto por computador, utilizando módulos com tecnologia *Xbee*.

SILVA, (2010) Expõe um sistema capaz de monitorar à distância ambientes insalubres ou de pequenas dimensões, que tornam o acesso humano perigoso ou praticamente impossível. Este sistema foi criado exatamente para evitar a presença humana nesses locais. O projeto consiste na utilização de um veículo e uma microcâmera para registrar imagens do ambiente a ser monitorado. Além da câmera, o carro é dotado de um sensor que registra a todo instante a temperatura do ambiente, um sensor de luminosidade responsável pela ativação dos faróis e um localizador sonoro avisando ao usuário o local onde o veículo se encontra, evitando a perda do equipamento. Também foi desenvolvida uma aplicação para controlar a câmera, ampliando assim, o campo visual a ser monitorado. Tanto o veículo como a microcâmera são controlados remotamente através de um computador.

COTTA, (2010) Apresenta o desenvolvimento de um sistema de controle remoto para um carro autônomo. Para tanto, foi disponibilizado um programa que permite operar o veículo ao mesmo tempo em que exibe dados coletados de sensores e vídeo em tempo real. Os veículos operados por controle remoto são usados em variadas situações, utilizando-se comunicação *wireless*, o que possibilita o acesso a informações sobre o veículo e o envio de comandos através de um computador externo. Assim os resultados alcançados neste projeto foram bastante satisfatórios, visto que, o controle do veículo pode ser realizado sem maiores dificuldades após um pequeno período de aprendizado com o sistema de operação remota.

PROCOPIO & VERISSIMO, (2010) Mostra o desenvolvimento de um dirigível *indoor* rádio controlado e todas as tecnologias envolvidas em sua construção. Para esse fim, foram utilizados conceitos de programação *desktop* (C#) e de micro controladores (PIC). O dirigível é basicamente constituído de: um balão inflado com gás hélio, uma estrutura de gôndola, uma câmera *wireless* e três motores de corrente contínua. Um dos motores controla o deslocamento do dirigível na direção horizontal, responsável pela propulsão. Um segundo

motor controla o dirigível na direção vertical, responsável pela elevação. O terceiro motor localizado na parte traseira do balão é utilizado como leme, para controlar a direção do voo. A câmera implantada na parte frontal do balão capta as imagens do ambiente.

Este projeto mostra o processamento de imagem utilizando o *Framework Aforge.NET* para obtenção e tratamento de imagens utilizando câmera acoplada ao dirigível. Com a utilização dessas ferramentas foi possível obter uma visão de situações perfeitamente aplicáveis na exploração das áreas de monitoramento de ambientes, sistema autônomo de navegação, entre outros.

VIEIRA, (2008) Aborda um sistema com técnica de banco de dados de imagem, onde apresenta um sistema que permita a publicação, a recuperação e a visualização de imagens em banco de dados, faz-se necessário a construção de diversos módulos e mecanismos para prover a recuperação de imagens mais condizente com a necessidade do usuário. Para que isto seja possível, as imagens precisam ser descritas antes de serem publicadas.

Neste trabalho foram realizados testes nos quais os resultados obtidos possibilitam a descrição de imagens com a finalidade de construir um banco de dados de imagens bastante funcional.

## **1.5 Organização do documento**

Esta monográfica esta organizada em seis capítulos da seguinte forma:

No capítulo 1, são explicados a motivação, definição do problema, objetivo do projeto, organização do documento e os trabalhos relacionados.

No capítulo 2, é apresentada a fundamentação teórica para auxiliar na elaboração e no entendimento e funcionamento do aplicativo desenvolvido.

No capítulo 3, é abordado o desenvolvimento do projeto e os requisitos necessários para que possa ser alcançado o objetivo esperado para a elaboração do aplicativo.

No capítulo 4, é apresentada a construção do aplicativo, e suas etapas. Além de demonstrar telas e trechos de códigos implementados.

No capítulo 5, é abordada a análise dos resultados alcançados com a construção deste projeto no que diz a respeito à recepção das imagens que são enviadas por uma câmera.

Por fim no capítulo 6 é apresentada a conclusão e os trabalhos futuros relacionados a esta monografia.

## Capítulo 2

### Fundamentação Teórica

Neste capítulo são descritas as referências bibliográficas para o entendimento da monografia, onde os temas tratados são conceitos básicos de imagem, robótica móvel, banco de dados, além de *framework Aforge.Net* e linguagem de programação C#.

#### 2.1 Robótica Móvel

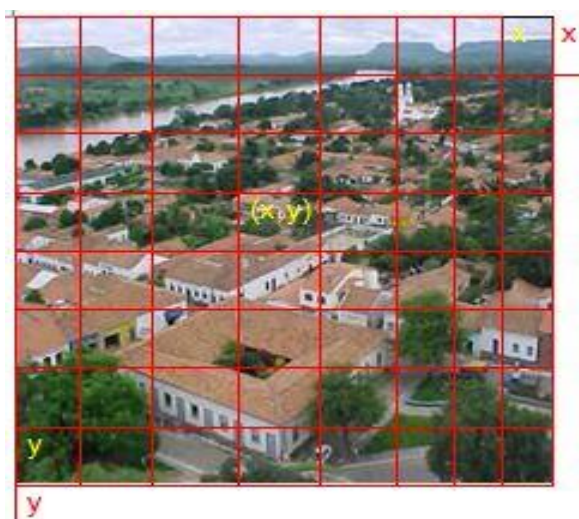
A robótica é uma parte da tecnologia que engloba mecânica, eletrônica e computação, que atualmente trata de sistemas compostos por máquinas e partes mecânicas automáticas e controladas por circuitos integrados, tornando sistemas mecânicos motorizados, controlados manualmente ou automaticamente por circuitos elétricos. O robô é um aparelho capaz de realizar determinada tarefa autônoma, através de recursos computacionais tais como microprocessadores e computadores acoplados. (VICTOR, 2010)

A evolução dos robôs tem ganhado nos últimos anos um amplo destaque junto à mídia é a sociedade de um modo geral. Atualmente as atenções se voltam para robôs móveis capazes de se deslocar no ambiente. A aplicação prática de robôs móveis em nossa sociedade vem comprovando o mercado promissor desta área. De acordo com (FERNADO e VALLE, 2009) o uso de robô móveis em aplicações (e.g. aspiradores de pó e cortadores de grama robóticos), industriais (e.g. transporte automatizado e veículos de carga autônomos), urbanas (e.g. transporte público, cadeiras de rodas robotizadas), militares (e.g. sistemas de monitoramento aéreo remoto – VANTs , transporte de suprimentos e de armamento em zonas de guerra, sistemas táticos e de combate) e de segurança e defesa civil e militar (e.g. controle e patrulhamento de ambientes, resgate e exploração em ambientes hostis), demonstram a grande gama de aplicações atuais dos robôs móveis.

Assim a robótica móvel vem cada vez mais se difundindo em nosso meio e exercendo inúmeras tarefas de risco e/ou que exigem do homem um grande esforço físico, além de explorar e monitorar ambientes. Neste contexto o projeto do aplicativo serve para auxiliar os robôs móveis na exploração de ambientes através de câmera.

## 2.2 Conceitos de Imagem Digital

A imagem digital tem a forma de uma matriz  $x$  e  $y$  de valores de pixel onde se refere ao uma função  $f(x, y)$  ilustrado na Figura 1, onde  $(x$  e  $y)$  denotam as coordenadas, uma imagem digital pode ser considerada como sendo uma matriz cujos índices de linhas de colunas identificam um ponto na imagem, e correspondem ao valor do elemento da matriz identificam-se os pixels da imagem. Portanto, podemos considerar que uma imagem digital pode ser representada por uma matriz  $(x$  e  $y)$  onde se atribui para cada elemento (pixels) dessa matriz o valor do seu respectivo ponto da imagem, sendo este valor positivo e identificando os níveis de cores de cada ponto que constitui a imagem. (ZACCARIOTTO, 2010).



**Figura 1. Matriz de uma Imagem Digital**  
**Fonte: Autoria própria.**

Na Figura 1 há amostragem da divisão do plano  $x, y$  onde a imagem é dividida em 8 linha e 8 colunas. Cada coordenada e intersecção de uma linha e uma coluna é um *pixels* que é transformado no formado matriz para diminuir o tamanho da imagem para o armazenamento no banco de dados.

Nesse contexto a imagem digital pode ser representada por uma matriz  $(x$  e  $y)$  ou um vetor. Cada elemento, (*pixels*) no projeto das imagens salvas no banco é convertido em bytes para serem salvas em forma de matriz deixando o banco de dados mais leve.

## 2.3 Banco de Dados

Os bancos de dados e sua tecnologia estão provocando um grande impacto no crescimento do uso de computadores. É viável afirmar que eles representam um papel crítico em quase todas as áreas em que são utilizados, incluindo negócios, comércio eletrônico, engenharia, medicina, direito, educação e ciências da informação. O banco de dados é uma coleção de dados, estes dados são fatos que podem ser gravados e que possuem um significado implícito. (ELMASRI & NAVATHE, 2005).

Resumido, o banco de dados garante a segurança e a integridade dos dados com eficiência, e estes serão armazenados de forma que possam ser recuperados e salvos quando necessário.

### 2.3.1 Banco de dados *Sql Server 2008*

O banco de dados *Sql Server* é um servidor mais abrangente e segura, confiável, gerenciável e escalonável para as aplicações de missão crítica, ao mesmo tempo em que permite que os desenvolvedores criem novas aplicações que possam armazenar e consumir quaisquer tipos de dados em qual qualquer dispositivo (MICROSOFT CORPORATION, 2012).

O *Sql Server 2008* permite que a intrigação com o Visual Studio 2010 com função de manipulação dos dados de forma rápida e de fácil construção do banco de dados, além de oferece o armazenamento das imagens nos mais diversos formatos(GIF,BMP,JPG, etc).

Resumido, no projeto utilizou-se o *Sql Server 2008* devido a sua integração com o Visual Studio 2010 e por ser este um banco de dados gratuito, além de permite que as imagens recebidas no computador sejam salvas de forma segura e por possuir uma ferramenta de design que integra o Visual Studio e que possibilita ao programador arquitetar o diagrama do banco de dados em ambiente gráfico.

# Capítulo 3

## Concepção Do Projeto

### 3.1 Introdução

Neste capítulo é abordado o desenvolvimento do projeto, os requisitos necessários para que possa ser alcançado o objetivo esperado e para a elaboração do aplicativo que auxilie o robô móvel na exploração de ambientes através de câmera sem fio. Além de apresentar as ferramentas utilizadas no desenvolvimento do aplicativo que serão imprescindíveis para a implementação do mesmo.

### 3.2 Análise dos Requisitos

Na análise de requisitos para a construção do aplicativo foi avaliada a necessidade de um software para a recepção e armazenamento de imagens enviadas por uma câmera acoplada a um robô móvel. No projeto de robótica móvel no Laboratório de Investigações e Pesquisas em Poéticas Digitais (LIPPO) observou-se inicialmente a necessidade de um aplicativo para exibir imagens transmitidas em tempo real do ambiente e também para armazená-las no computador do usuário.

### 3.3 Requisitos Funcionais

Os requisitos utilizados apresentam e descrevem o comportamento do aplicativo.

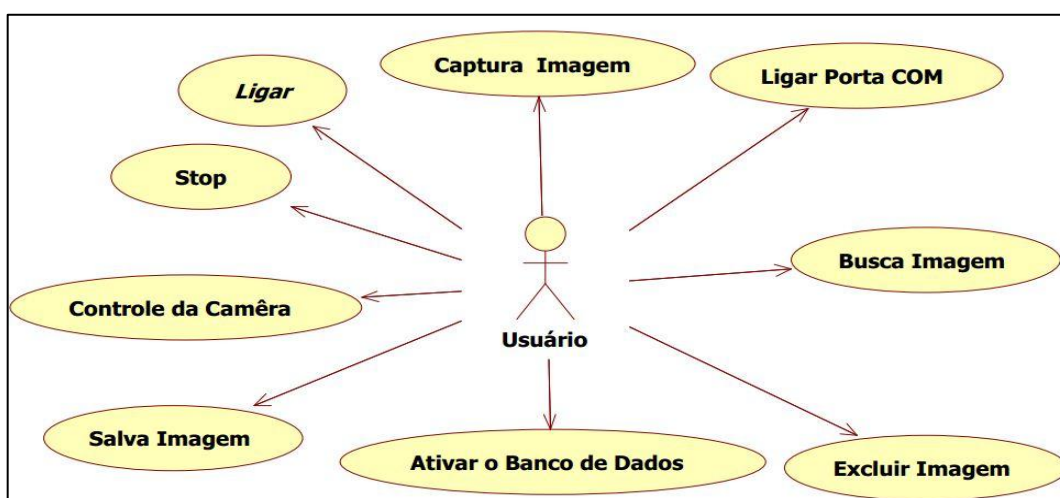
Analisando o objetivo do “ImageBot“, foram considerados os requisitos mais importantes para este projeto. A seguir são listados os requisitos funcionais para a elaboração e desenvolvimento do aplicativo:

- Armazenar as imagens capturadas no computador do usuário;
- Permitir a Visualização das Imagens em tempo real através da câmera sem fio;
- Permitir o usuário gerenciar as câmeras que estão conectadas na porta USB do computador;

- Capturar as imagens do ambiente que esta sendo monitorado;
- Permitir o controle do robô móvel remotamente através de comandos enviados por radiofrequência;
- Permitir ao usuário lista as porta COM disponível no computador;
- Possuir função de excluir, salva e busca a imagem no banco de os dados localizada no computador.

### 3.4 Caso de Uso

O caso de uso é a especificação de uma sequência de funcionalidade do sistema e os atores externos que utilizam esse sistema. Segundo (BEZERRA, 2007) “o caso de uso deve definir o uso de uma parte da funcionalidade de um sistema, sem revelar a estrutura e o comportamento interno desse sistema”. A Figura 2, ilustra o digrama de caso de uso do aplicativo “ImageBot”.



**Figura 2 - Diagrama de Caso de uso do Aplicativo (ImageBot)**  
 Fonte: Autoria própria.

A abaixo é apresentado cada funcionalidade do Caso de uso:

O caso de uso “Ligar”, o usuário dará um clique sobre o botão para exibir a imagem da câmera que esta conectada na porta USB do computador e irá exibir a imagem da câmera.

Caso de uso “Captura Imagem”, trata o botão que capta a imagem do ambiente para ser armazenada no banco de dados, o usuário irá dar um clique para executa a função.

Caso de uso “Ligar Porta COM”, trata-se do botão para ligar a porta do dispositivo



que este conectado no computador.

Caso de uso “Ativar o Banco de Dados”, o usuário irá clicar no botão do aplicativo “Ativar o Banco de Dados” ativando a conexão do banco de dados.

Caso de uso ”Salva Imagem”, o usuário ira clicar no botão para salva imagem que foi capturada do ambiente e armazenar no banco de dados.

Caso de uso “Excluir Imagem”, o usuário irá clicar no botão para excluir imagem que esta no banco de dados.

Caso de uso “Busca Imagem”, o usuário clicará sobre o botão para buscar no banco a imagem desejada.

Caso de Uso “Controle da Câmera”, o usurário ira envia comando de controles através das teclas pressionadas no computador.

Portando o caso de uso mostra todas as funcionalidades que o usuário pode fazer no aplicativo (ImageBot) como uma forma de interação facilitada.

### 3.5 Ferramentas Utilizadas

#### 3.5.1 Robô Móvel

O robô móvel utilizado nos testes foi desenvolvido pelo estagiário Thiago José Barbosa Lima e todas as atividades foram realizadas no LIPPO da Universidade Federal do Piauí, Campus de Picos.

A estrutura do robô é formada por uma placa micro controladora (Arduino), fios, resistores e sensores, além de duas rodas, dois motores e uma placa de circuito. Na figura 3 é apresentada a imagem do robô móvel desenvolvido.

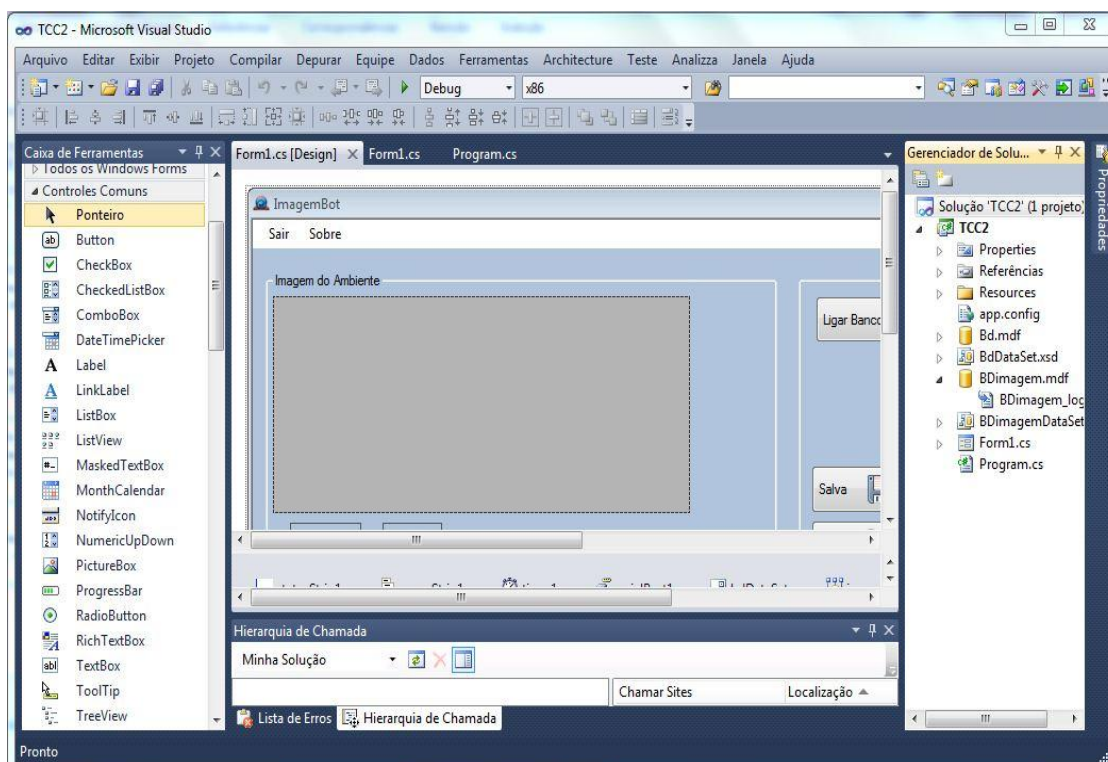


**Figura 3 - robô móvel**  
**Fonte: Autoria própria.**

### 3.5.2 Ambiente de Desenvolvimento

A Microsoft lançou oficialmente no dia 14 de abril de 2010 a versão do Visual Studio 2010, a qual traz diversas melhorias em todos os aspectos, focando a produtividade do desenvolvedor, melhorias no gerenciamento do ciclo de vida de uma aplicação, novidades de testes, novas funcionalidades, melhorias consideráveis no *Framework 4*. A IDE do Visual ES 2010 ficou muito mais fácil de entender, customizar e trabalhar. Isto tudo sem comentar sobre as melhorias nas linguagens C#, VB.NET 10, e a quantidade de novidades da IDE (MSDN,2012).

Para o desenvolvimento do projeto foi utilizada a plataforma ambiente de programação Microsoft Visual Studio 2010, usando a linguagem de programação C#. Devido a interação com o *Framework Aforge.Net* foi empregado o método de filtros de processamento de imagem e de vídeo, ideal para o desenvolvimento do aplicativo, por sua facilidade do uso de objetos, eventos, versatilidade para a construção da interface e a integração com o banco de dado *Sql Server*. Na Figura 4 é ilustrada uma das versões da Visual Studio 2010 utilizada para o desenvolvimento do aplicativo.



**Figura 4. Imagem da Interface do Visual Studio 2010**  
Fonte: Autoria própria.

### 3.5.3 Aforge.Net Framework

O *Aforge.Net* é de código aberto (*open source*), escrito em C# e é projetado para desenvolvedores e pesquisadores nas áreas de Visão Computacional e Inteligência Artificial. Atua diretamente nas áreas de processamento de imagens, redes neurais, algoritmos genéticos, aprendizado de máquina e robótica (ANDREW KIRILLOV). O *Framework* é composto por um conjunto de bibliotecas e aplicativos de exemplo, e tem várias bibliotecas que demonstram suas características:

- *AForge.Imaging* - biblioteca com rotinas de processamento de imagem e filtros;
- *AForge.Vision* - biblioteca de visão por computador;
- *AForge.Video* - conjunto de bibliotecas para processamento de vídeo;
- *AForge.Neuro* - Biblioteca de computação neural redes;
- *AForge.Genetic* - biblioteca de programação evolução;
- *AForge.Fuzzy* - biblioteca cálculos *fuzzy*;
- *AForge.Robotics* - Biblioteca de apoio de alguns kits de robótica;
- *AForge.MachineLearning* - Biblioteca de aprendizagem de máquina.

Assim, neste projeto utilizou-se a biblioteca *Aforge.Video* como parte principal devido a biblioteca possuir classes diferentes, que fornecem acesso a dados de vídeo utilizado no aplicativo (ImageBot).

### 3.5.4 Linguagem de Programação C#

O C# é uma linguagem de programação criada para o desenvolvimento de uma variedade de aplicações que executam sobre o Framework.NET. A linguagem C# é simples, poderosa, com tipagem segura e orientada a objetos. As várias inovações no C# permitem o desenvolvimento rápido de aplicações, mantendo a expressividade e a elegância das linguagens C. (MICROSOFT CORPORATION, 2012).

A implementação da linguagem C# pela Microsoft. Visual Studio suporta Visual C# com um editor de código completo, compilador, modelos de projeto, designers, assistentes de código, um depurador poderoso, de fácil uso e outras ferramentas. O *Framework.Net* fornece acesso a serviços de operação do sistema e outros, além de acelerar o ciclo de desenvolvimento de forma significativa (MICROSOFT CORPORATION, 2012).

A linguagem C# suporta os conceitos de encapsulamento, herança e polimorfismo, e todas as variáveis e métodos, incluindo o método principal (Main). O ponto de execução de uma aplicação é encapsulada em definições de classes. Uma classe derivada pode herdar diretamente somente de uma classe pai, mas pode herdar qualquer quantidade de interfaces. Os métodos da classe derivados que substituem métodos virtuais de uma classe pai exigem a utilização da palavra-chave *override* como forma de evitar a redefinição acidental. Em C#, uma *struct* é como uma classe simplificada; é um tipo alocado em pilha que pode implementar interfaces, mas não suporta herança. (MICROSOFT CORPORATION,2012).

Esta sessão apresenta os conceitos básicos de linguagem C# e também as características da linguagem e suas funções. A escolha da linguagem se deu pela facilidade da interação com as bibliotecas do *Aforge.Net* para fazer o processamento da imagem e do vídeo.

# Capítulo 4

## IMPLEMENTAÇÃO

Neste capítulo será apresentada a descrição do desenvolvimento do aplicativo, as ferramentas utilizadas na implementação e as tecnologias de programação utilizadas.

### 4.2 Descrição do Software

#### 4.2.1 Interface do Aplicativo

A interface do “ImageBot” na tela concentra todas as interações com o usuário, apresentando as funções para que este interaja com o programa e possa realizar o controle do robô móvel e o armazenamento das imagens capturadas. Ao iniciar este aplicativo, o usuário visualizará a sua interface principal, contendo todas as funções necessárias para iniciar a captura da imagem transmitida pela câmera sem fio.

O aplicativo está dividido em cinco painéis onde temos: Imagem do Ambiente, Controle da Câmera, Controle do Robô Móvel, Conexão e Banco de Dados, como mostra a Figura 5. Além de possuir uma barra de ferramenta para que o usuário possa acessar a ajuda. E uma barra de status que apresenta algumas informações sobre as operações efetuadas pelo usuário.



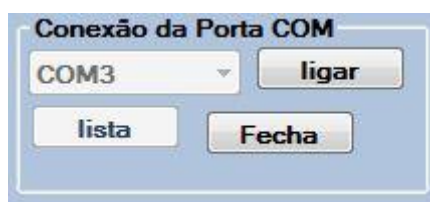
Figura 5 - Interface Principal do Aplicativo  
Fonte: Autoria própria.

A Figura 6 exibe o painel de imagem do ambiente, neste painel existem quatro funções: O botão “ligar” que ativa a câmera de vídeo conectada e capta a imagem do ambiente exibindo-a na tela do aplicativo em tempo real, o segundo botão de “stop” que para a execução do vídeo, o terceiro botão com o nome “captura”, capta a imagem que esta sendo exibido na tela, o quarto botão com o nome “lista de dispositivos” lista os dispositivos que estão conectados na porta USB do computador onde o usuário escolhe a câmera que esta conectada.



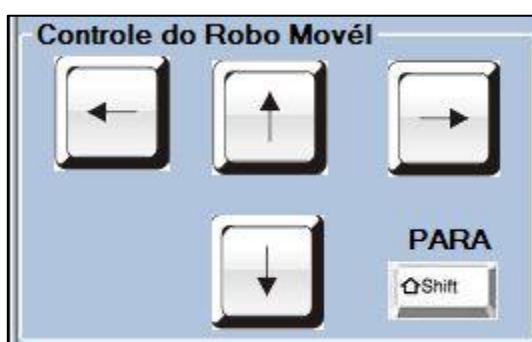
**Figura 6 - Imagem do Painel Imgem do Ambiente**  
**Fonte: Autoria própria.**

A Figura 7 exibe o painel de configuração da conexão porta COM do computador. Este painel é bastante simples e contém apenas três botões. O botão com nome “lista” que trata da abertura da porta COM de todos dispositivos conectados ao computador onde enviar dados por intermédio dessa porta, o segundo botão “ligar” faz com que a porta seja ativa para o uso dos dispositivos ligados na porta USB do computador, o terceiro botão com nome “Fecha” representa o fechamento da porta que esta ativada.

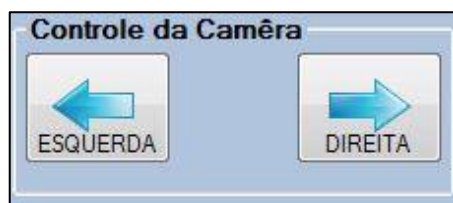


**Figura 7 - Painel de Conexão de Porta COM**  
**Fonte: Autoria própria.**

As Figuras 8 e 9 apresentam as interfaces do aplicativo que fazem os controles do robô móvel e da câmera acoplada. Quando pressionada a setas do computador são enviados comandos para controlar o robô móvel através da conexão do módulo Xbee, conectado na porta USB do computador, os sinais são transmitido até o outro módulo acoplado ao robô, assim fazendo o controle remotamente do aplicativo. Na Figura 8 vemos as setas que indicam as direções do robô, quando pressionadas mandam os comandos direita, esquerda, frete e trás, respectivamente para o receptor no robô móvel. A Figura 9 mostra a seta esquerda e direita para o controle da câmera quando estas são pressionadas.



**Figura 8 - Painel do Controle do Robô Móvel**  
Fonte: Autoria própria.



**Figura 9 - Painel do Controle da Câmera**  
Fonte: Autoria própria.

Na Figura 10 vemos o painel Banco de Dados, este painel é um dos principais, pois nele estão todas as funções para a manipulação do banco de dados, e contem quatro botões:

- Botão com o nome “Ligar Banco De Dados” este faz a conexão com banco de dados *Sql Server* 2008, presente no pacote de instalação do Visual Studio onde é executado localmente no próprio computador. Este botão quando pressionado, ativa os botões Salva, Busca, Excluir e ativa o conexão com banco de dados exibido ao lado todas as imagem com as informações que já foram salvas.

- Botão com a identificação “Salva” quando pressionado, salva as imagem capturadas. É necessário preencher o campo da descrição para que sejam salvas no banco de dados. Desta forma a imagem só será salva se o campo da descrição estiver preenchido.
- Botão com nome “Busca“ serve para buscar uma imagem no banco de dados quando informado ID da imagem. Quando pressionado “Busca” no banco se o ID existir é exibida na tela a imagem desejada, caso não tenha é mostrada uma mensagem informando que imagem não existe.
- O Botão “Excluir“ faz com que a imagem seja excluída do banco de dados, quando pressionado é excluída pelo ID informado.



**Figura 10 - Painel do Banco de Dados no Aplicativo**  
Fonte: Autoria própria.

Além dos Botões, há dois campos, um para digitar o código da imagem, outro para digitar a descrição da imagem, e também são exibidos os dados que estão armazenados no banco de dados. Portanto o aplicativo permiti ao usuário salvar, buscar e excluir imagens no banco de dados, além de visualização individual das imagens.



## 4.2.2 Arquitetura de Comunicação

Esta arquitetura representa a comunicação do aplicativo (ImageBot) instalado no computador com o robô móvel onde a transmissão da imagem é feita através da câmera sem fio que esta acoplada ao robô móvel. Com o aplicativo executando no computador o usuário consegue visualizar as imagens capturadas pela câmera e enviar comandos para o controle do robô através dos módulos *Xbees*. O sistema de comunicação aplicação/robô móvel será implementado na forma descrita na Figura 11 abaixo.

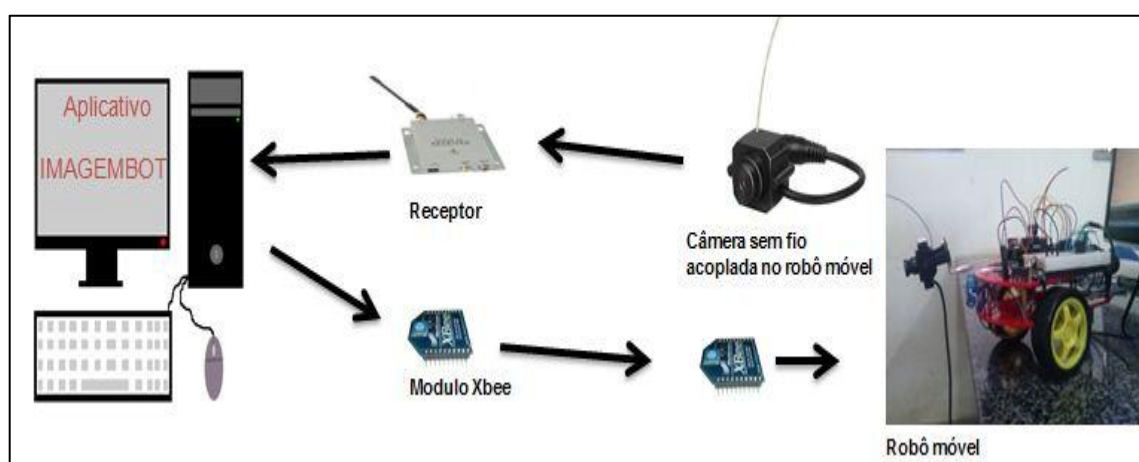





Figura 11 - Arquitetura de Comunicação  
Fonte: Autoria própria.

## 4.2.3 Biblioteca Aforge.Video.DirectShow

A utilização do *Aforge.Video*, no projeto foi devido as classes que permitem ter acesso a fontes de vídeos usando *DirectShow*, escrita em C#. O *Directshow* tem como objetivo principal permitir o desenvolvimento do aplicativo, pois tem uma enorme variedade de filtros multimídia como criar efeitos de vídeos e áudio, frame-rate automático e ainda outros. Na Figura12 são exibidas as da classe da biblioteca *Aforge.Video.DirecShow* utilizados na implementação do “ImageBot” e faz o processamento dos vídeos e imagens capturadas. No projeto foram utilizadas as classe *FilterInfoCollection*, *VideoCaptureDevice* e a *FilterInfo*.

	Classe	Descrição
	<a href="#">FileVideoSource</a>	Fonte de vídeo para arquivos de vídeo.
	<a href="#">FilterCategory</a>	Categorias DirectShow filtro.
	<a href="#">FilterInfo</a>	DirectShow informações do filtro.
	<a href="#">FilterInfoCollection</a>	Coleção de objetos de informação dos filtros.
	<a href="#">VideoCapabilities</a>	As capacidades do dispositivo de vídeo como o tamanho ea taxa de frame.
	<a href="#">VideoCaptureDevice</a>	Fonte de vídeo para dispositivo local de captura de vídeo (por exemplo USB webcam).
	<a href="#">VideoCaptureDeviceForm</a>	Local vídeo formulário de seleção de dispositivo.
	<a href="#">VideoInput</a>	Entrada de vídeo de uma placa de captura.

**Figura 12 - Classe do Aforge.Video.DirectShow**  
**Fonte: Autoria própria.**

#### 4.2.4 Aquisição do Vídeo

Na aquisição do vídeo foram implementadas as funções para fazer a conexão com a câmera sem fio, e sua conversão a dados binários, onde primeiramente um fluxo de vídeo é capturado e em seguida processado quadro a quadro da imagem. As subseções abaixo descrevem os métodos e funções. Maiores detalhes sobre o código implementado podem ser vistos no Apêndice I onde está a descrição do algoritmo.

##### 4.2.4.1 Método *BuscaDispositivo*

Este método implementa a função para listar os dispositivos locais (conectado diretamente na porta USB do computador) utilizando o *FilterInfoCollection* para enumerar os dispositivos de Vídeo . A Figura 13 mostra um trecho do código implementado no algoritmo do aplicativo. No Apêndice I está o código completo da aplicação onde há maiores detalhes sobre o método *BuscaDispositivo*.

```
public void BuscaDispositivo(FilterInfoCollection Dispositivos) // metodo para lista os dispositivos do usb
{
    // Enumerar dispositivos de video
    for (int i = 0; i < Dispositivos.Count; i++) //lista
        proDispositivos.Items.Add(Dispositivos[i].Name.ToString());
    proDispositivos.Text = proDispositivos.Items[0].ToString();
}
```

**Figura 13 - Trecho do Código de BuscaDispositivo**  
**Fonte: Autoria própria.**

#### 4.2.4.2 Classe VideoCaptureDevice

Esta classe foi utilizada no projeto devido à biblioteca *Afoge.Net* com a função principal para os quadros (frames) obtidos, este código implementado serve para captar dados de vídeo a partir do dispositivo local de captura de vídeo, como a câmera sem fio, placa de captura ou qualquer coisa que suporta *DirectShow* interface. Na Figura 14 abaixo é mostrado um trecho do código implementado para fazer o processamento do vídeo que seja exibido na tela do aplicativo utilizando o evento *NewFrameEventHandler* para fazer a manipulação com quadros de vídeos. A Figura 14 trás também um trecho do código implementado que ligar a câmera que esta conectada no computador e depois exibi o vídeo na tela do aplicativo. A outra parte do código pode ser visto no Apêndice I.

```

private void Video_Click(object sender, EventArgs e)
{
    if (TemDispositivos) //dispositivos que esta conectado
    {
        //Fonte de vídeo para dispositivo local de captura de vídeo (por exemplo USB webcam).
        EntradaVideo = new VideoCaptureDevice(DispositivosDeVideos[proDispositivos.SelectedIndex].MonikerString);

        // Define novo manipulador de eventos de Frame, se precisamos de processamento de novos quadros
        EntradaVideo.NewFrame += new NewFrameEventHandler(CLONE_NovoFrame);

        // método deve ser usado para iniciar a reprodução da fonte de vídeo.
        EntradaVideo.Start();
        proDispositivos.Enabled = false;
        groupBox1.Text = DispositivosDeVideos[proDispositivos.SelectedIndex].Name.ToString();// adicionar na lista o dispositivo
        Status(true);
    }
}

```

**Figura 14 - Código para Captar Dados de Vídeos**  
Fonte: Autoria própria.

#### 4.2.4.3 Classe para Clonar a Imagem do Vídeo

Nesta etapa da implementação foi utilizado o filtro da classe *Aforge.Video* o método *NewFrameEventArgs* que tem argumentos para o evento novo quadro da fonte de vídeo. Na Figura 15 abaixo esta um trecho do algoritmo que demonstra o tratamento do quadro de imagem recebido da câmera. No instante da captura da imagem é clonada através da classe *Bitmap* que cria uma copia exata da imagem a ser exibida no *pictureBox1* do aplicativo.

```

private void CLONE_NovoFrame(object sender, NewFrameEventArgs eventArgs)
{ //Manipulador de eventos novo quadro

    Bitmap image = (Bitmap)eventArgs.Frame.Clone();
    //Inicializa uma nova instancia da classe Bitmap a partir da imagem existente especificado.
    pictureBox1.Image = image;
    // Clone () Cria uma copia exata desta imagem recebida da camera . (Herddado de Imagem ).
}

```

**Figura 15 - Trecho do Código Implementado para Clonar Quadro da Imagem**  
**Fonte: Autoria própria.**

#### 4.2.5 Banco de Dados BDIimagem

O Banco de dados BDIimagem foi desenvolvido no *Sql Server 2008* para a comunicação entre o aplicativo. Foi criada uma tabela de imagem que é responsável por armazenar a imagem, a descrição, o código identificador(Id) que permiti ao usuário gravar, ler e excluir imagens no Banco BDIimagem.

No aplicativo foi implementação a Classe *SqlConnection* para conectar com banco de dados. Esta classe contem vários propriedades e métodos para manipular a conexão com banco BDIimagem. Na Figura 16 é exibido um trecho do código implementado para fazer a inserção dos dados, utilizando o objeto *SqlCommand* que serve para executar comandos (*Insert, Select, Delete*) no banco.

```

this.sqlcmd = new SqlCommand();
sqlcmd.Connection = conexaoSQLServer;// fazer a conexão
if (sqlcmd.Parameters.Count == 0)// condição para ver se esta conectado
{ // no CommandText so SqlCommad,passo aintrução SQL referencia a inserção dos dados
    this.sqlcmd.CommandText = "INSERT INTO Imagens(descricao,imagem) values(@descricao,@imagem)";
    // Parâmetro da descrição para ser inserido no banco
    this.sqlcmd.Parameters.Add("@descricao", System.Data.SqlDbType.VarChar, 150);
    this.sqlcmd.Parameters.Add("@imagem", System.Data.SqlDbType.Image);
}

```

**Figura 16 - Trecho do Código Para Fazer a Inserção da Imagem**  
**Fonte: Autoria própria.**

#### 4.2.5.1 Processamento da Imagem em Bytes

Para gravar e recupera imagens no banco de dados BDIImagem foi implementada uma classe Byte que converte as imagens em bytes, pois estas não pode ser armazenado diretamente no banco de dados. A Figura 17 mostra um trecho do algoritmo que converte a imagem em Array de bytes e inseri os dados em bytes no banco BDIImagem e converte os dados em bytes novamente para o formato imagem quando o usuário busca a imagem.

```

    •
    •
    •
public static Byte[] Codifica(Image foto)
{ // contém os bytes da imagem PictureBox
  MemoryStream memoria = new MemoryStream();
  // classe MemoryStream ler dados no fluxo de memória, para converte para dados bytes
  foto.Save(memoria, ImageFormat.Jpeg); // save imagem em formato Jpeg
  Byte[] MyData = new Byte[memoria.Length];
  /// / Converte a entrada em saída bytes.
  MyData = memoria.ToArray();
  // Escreve a segunda seqüência para o fluxo, byte do array.
  return MyData;
}
    •
    •
    •

```

Figura 17 - Trecho do Código para Converte Dados em Bytes  
Fonte: Autoria própria.

#### 4.5.4 Envio de Comandos

O envio de comando é realizado com a utilização da função *ProcessCmdKey* que é responsável por captura o código da tecla que esta sendo pressionada pelo usuário para o envio do sinal de controle para robô móvel. O método *KeyData* (Figura 21) compara se uma tecla foi pressionada ,as teclas *Left*, *Right*, *Up* e *Right* representam as direções: esquerda, direita, frente e trás, respectivamente.

Para enviar o comando pela porta COM é utilizado o método *serialPort.Write* que envia a *String* através do módulos *Xbee* para fazer a comunicação do aplicativo com o robô móvel. Esse método é mostrado na Figura 18.

```
protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
{
    // verifica se a tecla esta pressionada
    // é seta para esquerda
    if (keyData == Keys.Left)
    {
        serialPort1.Write("4");//hexadecimal 26
        pictureBox4.Image = Properties.Resources.right_up;
        pictureBox3.Image = Properties.Resources.esquerda;
    }
    else
    if (keyData == Keys.Right)// direita
    {
        serialPort1.Write("6");//hexadecimal 26
        pictureBox3.Image = Properties.Resources.left_up;
        pictureBox4.Image = Properties.Resources.direita;
    }
}

```

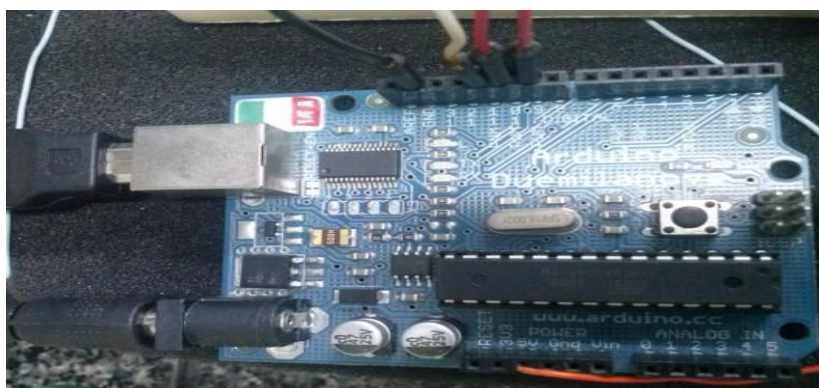
**Figura 18– Código da função que enviar comando pra robô móvel**  
**Fonte: Autoria própria.**

### 4.3 Descrição do Hardware

Essa sessão abordará os materiais utilizados para a montagem dos testes exibido no capítulo 5 onde foi utilizado dois módulos *Xbees*, câmera sem fio, receptor, Placa EASYCAP e Arduino Duemilanove.

#### 4.3.1 Arduino Duemilanove

O Arduino utilizado foi ATmega168 com clock de 16MHz. A placa tem 14 pinos digitais de entrada e saída e 6 pinos de entrada analógico. Possui 32 KB de memória para armazenar o código. Utilizando módulo Xbee acoplado no arduino por um para placa de conexão (Shield) para receber dos dados enviados do computador através dos módulos Xbee que é utilizado para fazer o controle do robô móvel. Na Figura 19 e apresentado o Arduino.



**Figura 19 - Arduino Duemilanove**  
**Fonte: Autoria própria.**

### 4.3.2 Módulos Xbee

Os módulos Xbee representados na Figura 20 foram acoplados um no arduino para receber os sinais enviados do computador para controlar a câmera acoplada ao robô móvel, o outro *Xbee* é responsável por receber os dados enviados do aplicativo pelo transmissor e passar os sinais para o arduino, o qual interpreta os dados e executa a ação correspondente. O transmissor (Xbee) está conectado na porta USB do computador onde os comandos são enviados do aplicativo para executar o controle do robô móvel.

O protocolo utilizado pelos módulos é o IEEE 802.15.4, os módulos possuem antena XB24-AW-001, a frequência da operação é 2.4 Ghz, possuem alcance 100 metros em campo aberto.



Figura 20 –Módulos Xbee.

Fonte: Autoria própria.

### 4.3.3 Câmera sem Fio

Para captura das imagens no ambiente foi utilizada uma câmera de pequena dimensão instalada no robô móvel. Na Figura 21, é exibida a câmera sem fio para transmissão da imagem.



Figura 21 - Câmera sem Fio

Fonte: Autoria própria.

A câmera utilizada no projeto para visualizar o ambiente e captar as imagens, foi acoplada ao veículo. A transmissão do Vídeo da câmara é feito através de radiofrequência na faixa de 1,2 GHz, os sinais são transmitido através de uma antena onde são geradas ondas eletromagnéticas que se propagam no ar. As ondas são captadas pelo receptor que está conectado no computador através de uma placa de captura de vídeo denominado EASYCAP, a placa dispõe de uma saída USB e possui uma entrada de vídeo e duas de áudio, a conexão com o computador é feita através de uma porta USB, o receptor converte as imagens para o padrão USB para serem adquiridas facilmente em qualquer computador. Na Figura 22 visualizamos o receptor e placa EASYCAP, utilizados na captação dos sinais de vídeos.



**Figura 22 - Placa EASYCAP e Receptor da Câmera sem Fio**  
**Fonte: Autoria própria.**



## Capítulo 5

### Análise Dos Resultados

Este capítulo apresenta os resultados alcançados após implementação dos testes realizados no projeto, a fim de verifica se o objetivo foi obtido.

#### 5.1 Instalação do Executável

O aplicativo é executável em um pacote com todos os arquivos necessários para o seu funcionamento. Para uso do aplicativo deve ser executado o arquivo chamando TCC2 que se encontra dentro da pasta, conforme exibido na Figura 23.

TCC2.vshost.exe.manifest	130077559165471373	130066421290756809	130056988641789152	130036191621386281
TCC2.vshost.exe	130077538565457211	130066421216752576	130056927274109152	130036191339726281
TCC2.vshost	130077538554537192	130066420729784723	130056927218659152	130036191245186281
TCC2	130077538543773173	130066420686922271	130056910592879152	128177969230318299
TCC2.exe	130077538039112286	130066420634359265	130056847451279152	128177969127752433
BDimagem_log	130077533048720265	130066420584646422	130056847390119152	128177968300205100
BDimagem	130074979232914181	130066390340646564	130056846859319152	128177954687665782
Bd_log	130072494741361284	130066390106293160	130056008701810946	128177939866463270
Bd	130072494632061284	130066389949554195	130055996512920946	
AForge	130069993595204107	130066347044420161	130055987631520946	
AForge.Video	130069993595480107	130066347020908817	130055161808377395	
AForge.Video.dll	130069982193034107	130066346797716051	130055149235928292	
AForge.Video.DirectShow	130069982155714107	130066346252484865	130055149109281049	
AForge.Video.DirectShow.dll	130069982101644107	130066346169800136	130047314762008657	
AForge.dll	130069982020994107	130065688297246771	130047313170825761	
	130069981981314107	130065687984586771	130047313114722552	
	130069981961554107	130065687938456771	130047309662645105	
	130079321942053527	130068903357329154	130065687897066771	130044820485938143
	130077573244649511	130068903229199154	130065686514036771	130044820025991835
	130077573160565364	130068896175719154	130065686427156771	130044684062426135
	130077573143561334	130068889000219154	130065684870366771	130039347260922913
	130077573119537291	13006888980309154	130065684525276771	130038537473540993
	130077573110489276	130068850224399154	130065684470386771	130038517722610993
	130077573096605251	130068846145369154	130060337170984025	130038517609310993
	130077572956849008	130068846086139154	130057048736249152	130038486534240993
	130077572898972907	130068814581829154	13005704777989152	130036252425486281
	130077572849520820	130068814502719154	130056997836109152	130036219487466281
	130077571807543094	130068813951259154	130056988714089152	130036219387786281
	130077571660264670	130068813896949154	130056988714089152	130036219346946281
	130077562168289014			

**Figura 23 - Pacote com os arquivos necessários para execução do aplicativo**  
**Fonte: Autoria própria.**

Os resultados obtidos com o aplicativo executável não ocorreram nenhum problema, no qual pode ser executado em outro computador utilizando a pasta com os pacotes do aplicativo. No entanto, o código escrito em linguagem C# não pode ser executável em outro sistema operacional pelo fato de o sistema de diretório adotado ter sido o padrão Windows 7 Ultimate 32 bits, com processador Intel Core 2 de 2.0 Ghz e memória de 3 Gb Ram .

## 5.2 Avaliação das Funções do Aplicativo

Na primeira avaliação do aplicativo (ImageBot), utilizou-se uma primeira versão que não tinha funções para controlar o robô móvel tendo em vista uma necessidade de controlar o robô móvel e a câmera, foram adicionados às duas funções no aplicativo de controle do robô.

O funcionamento geral do aplicativo (ImageBot) é bem simples, o aplicativo recebe o fluxo de vídeo da câmera instalada no robô móvel. O vídeo é transmitido por rádio frequência a o receptor conectado a entrada de uma placa de captura de vídeo no PC. A partir dessa imagem captura é armazenado no banco de dados com tamanho de (392x182)pixels para depois ser análise da imagem. A comunicação para controlar o robô móvel ocorre por meio do módulo Xbee de rádio frequência. O computador contendo um aplicativo instalado envia os comandos para receptor acoplado no robô móvel os sinais que acionam controle do robô móvel.

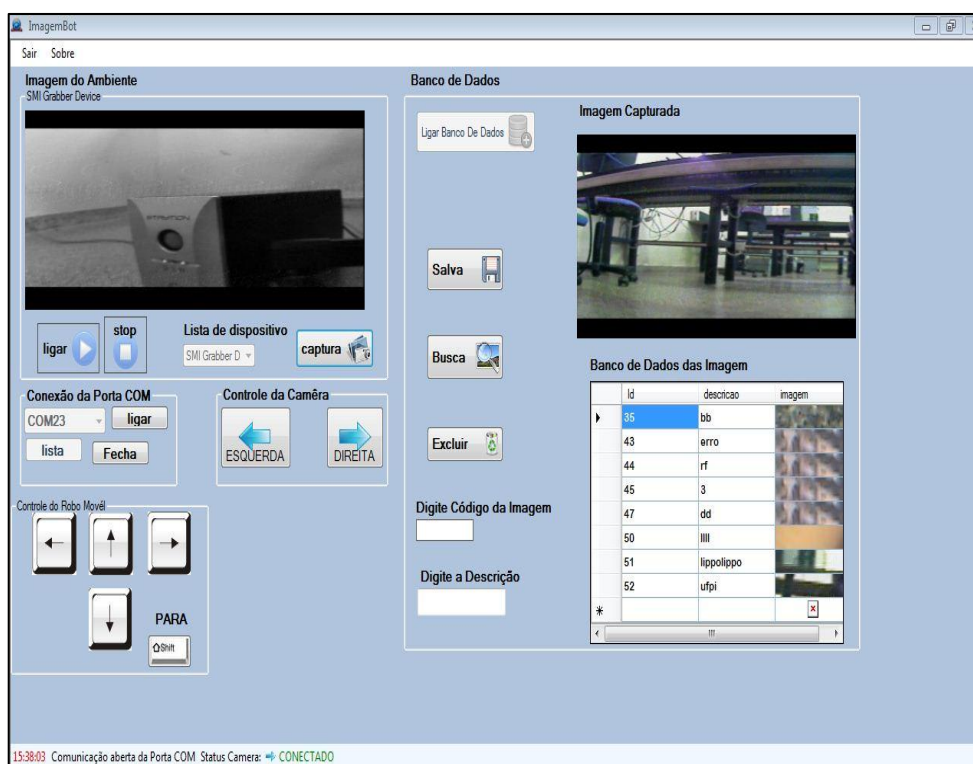
O desempenho da aplicação é muito dependente da câmera sem fio utilizada de acordo com a câmera utilizada à qualidade da imagem exibida na tela podem ser da baixa resolução ou alta. No mercado hoje esta disponível micros câmera que possui alta definição de imagem e compensação automática de luz, ou seja, ajusta de acordo com a variação do tempo.

## 5.3 Ambiente de Teste

Os testes foram realizados na Universidade Federal do Piauí - UFPI, do Compus de Picos-PI no LIPPO (Laboratório de Investigações e Pesquisas em Poéticas Digitais). Na realização dos testes do aplicativo foram utilizados os seguintes equipamento: um notebook positivo Intel Core 2 Duo 2.0 GHz com memória RAM de 3GB , o Sistema operacional Windows 7. Além ser utilizado uma robô móvel desenvolvido no LIPPO na realização do testes .

## 5.4 Execução do Aplicativo

Na execução o aplicativo (ImageBot) listou todos os dispositivos de vídeo que estavam conectados na porta USB do computador. O aplicativo listou todos os dispositivos capturou a imagem através da webcam nesse parte de teste foi satisfatório. Na parte de ativar o banco de dados o aplicativo levou aproximadamente 3 segundos em vários testes realizados, para exibir o painel do Banco de dados da Imagem além de para todos as funções do aplicativo nesse tempo de 3 segundos depois disso o aplicativo volta a funcionar normalmente. Na Figura 24 e apresentado a imagem da execução do aplicativo.



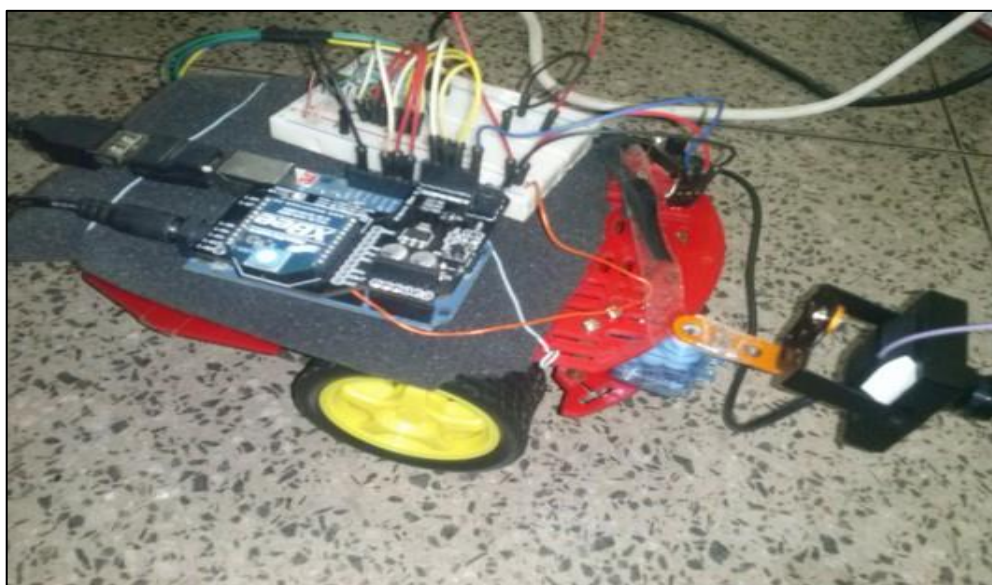
**Figura 24 - Imagem da Execução do Aplicativo na Captura da Imagem e o Armazenamento**  
**Fonte: Autoria própria.**

## 5.5 Teste com Hardware

Na realização de teste com câmera sem fio foi identificado um problema na transmissão do vídeo para computador devido interferência. Toda vez que era transmitido o vídeo ficava com interferência devido baixa qualidade de transmissão da antena. Além da interferência foi encontrada a dificuldade para fazer alimentação da câmera, pois não tinha baterias adequadas para fazer a alimentação de energia.

O robô móvel partiu do LIPPO de um ponto próximo ao computador até em ambiente aberto no Corredor da UFPI até distância aproximada de 15 metros, os itens funcionaram de perfeitamente. Ao passar de essa distância o receptor deixou de receber as imagens da câmera sem fio. Isso era já esperado devido à baixa qualidade da transmissão da câmera sem fio, pois de acordo com o manual pode chegar a uma distância de 15 a 30 metros de alcance. Após o término do teste para verificar o alcance da câmera sem fio, foi executado o teste com o módulo *Xbee* não teve problema, pois o robô móvel recebeu os sinais de controle aproximadamente até 70 metros de acordo com o manual do fabricante. Portanto os testes com o *Xbee* ocorreram de acordo com o planejado.

Na figura 25 apresenta a foto do robô móvel com os seus componentes acoplados ao mesmo.



**Figura 25 - Robô Móvel Desenvolvido no LIPPO, utilizado para os testes.  
Fonte: Autoria própria.**

## 5.6 Teste do Banco de Dados

Nos testes realizados foi identificado um atraso de 3 segundos na hora de ativar a conexão esses 3 segundos foi baseado no relógio da aplicação. No teste realizado de incluir, excluir e consulta as imagem no banco de dado BDimagem não foi percebido erro. Para salva imagem de tamanho (392x182 ) pixels o usuário só precisar informa a descrição, pois a o campo do código é incrementado automaticamente na tabela Imagem. Na busca da imagem precisar informa o código para exhibi na tela do aplicativo (ImageBot) a imagem que esta armazenada no banco de dados.

## 5.7 Dificuldades

As dificuldades encontrada para a implementação da monografia foi na utilização dos hardwares para a realização dos testes. As imagens capturas pela câmera sem fio foram exibidas com abaixa qualidade no aplicativo. A possível causa é a baixa qualidade do sinal da câmera sem fio utilizada na transmissão da imagem.

A fonte de alimentação foi insuficiente para realização dos testes, pois a câmera sem fio precisar de uma alimentação 12 volts. Na realização do teste com câmera foi realizado a alimentação com bateria de 9 volts que durou aproximadamente 10 minutos ligado.

Outra dificuldade foi a respeito a grande quantidade de conceitos de processamento de imagem que precisaram ser estudadas e compreendidas para utilização do *Aforge.Net* na implementação no aplicativo para captura a imagem da câmera.

## 6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foi desenvolvido um aplicativo para recepção de imagem capturada por uma câmera acoplada em robô móvel em tempo real, capaz de monitora ambiente. O aplicativo foi entregue com banco de dados *Sql Server* 2008 e utiliza a biblioteca *Aforge.Net* para processamento do vídeo e da imagem implementado na Linguagem *C#*.

O projeto atingiu o objetivo que era a recepção da imagem enviada por uma câmera sem fio com sucesso. O único problema foi à qualidade de transmissão da câmera, pois transmite em distância pequena e com baixa resolução da imagem.

Assim os resultados obtidos no projeto foram satisfatórios, o aplicativo pode ser usado para fazer monitoramento de ambiente perigoso e captura as imagens do ambiente no qual o robô móvel este incluído. Esse projeto pode trazer benéficos em projetos de robô móvel. Além disso, os alunos que tiverem interesse em dar continuidade a este trabalho, poderão incluir técnica de processamento de imagem para reconhecer os objetos do ambiente através da imagem.

### Trabalhos Futuros

Varias pesquisa podem ser criadas a parti desse trabalho aqui apresentado. Os itens propostos a seguir visam à expansão deste trabalho:

- Melhorias na interface para exibir mais na tela mais Imagens de viárias câmeras.
- Propor método de processamento de imagem para identificar objeto do ambiente através da câmera.
- Implementação de outra forma para se conecta com aplicativo como através de monitoramento via internet.
- Utilização de câmera com maior definição.
- Desenvolvimento de um sistema de autenticação para acesso seguro do banco de dados.

## REFERÊNCIAS

ANDREW KIRILLOV. **AForge.NET framework de código aberto**. Disponível em: <<http://code.google.com/p/aforge/>>. Acesso em: 17 dez. 2012.

NVATHE & ELMASRI, 2005 B.NAVATHE, S.; ELMASRI, R. **Sistema de Banco de Dados** : Tradução de Marília Guimarães Pinheiro 4 ed. Revisão Técnica de Luis Ricardo de Figueiredo. 4. ed . São Paulo: Pearson Education , 2005. 796p.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 3. ed. Rio de janeiro : Campus, 2007. 290p.

CAVALCANTE, M. U. **Sistema De Monitoramento com Câmera de Vídeo de Baixo Custo em Tempo Real**. 2011. 55f. Monografia (Bacharelado em Engenharia De

Teleinformática)UniversidadeFederal

DoCeará,Fortaleza,2011.Disponívelem:<[http://www.codeproject.com/Articles/16859/AForge-NET-open-sourceframework&usg=ALkJrhjGfz9VCE\\_nrBF3sAb3HO\\_LCa\\_o\\_w](http://www.codeproject.com/Articles/16859/AForge-NET-open-sourceframework&usg=ALkJrhjGfz9VCE_nrBF3sAb3HO_LCa_o_w)> .Acesso em: 02 jan.2013.

CESAR, Caio. **Projeto De Um Veículo Subaquático Autônomo** 2002.Disponívelem:<<http://www.maua.br/arquivos/index/h/b0a2b3c5feb0209562a0f8950078fd89>> . Acesso em: 11 dez.2012.

COTTA, P. S. **Arquitetura Cliente/Servidor para o comando remoto**. 2010. 42f. Monografia (Bacharelado em Engenharia de Controle e Automação) - Universidade Federal de Minas Gerais, Belo Horizonte, 2010. Disponível em: <<http://support.microsoft.akadns.net/kb/317701/pt-br>> .Acesso em: 05 fev. 2013.

MICROSOFT CORPORATION. Linguagem de Programação C#. Disponível em:<<http://msdn.microsoft.com/pt-br/vstudio/hh388566>> Acessado em:2 jan.2013.

MSDN. **Visual Studio 2010**. Disponível em: <<http://msdn.microsoft.com/pt-br/library/ff978714.aspx>> .Acessado em: 10 dez.2012.

FERNADO.D.W e VALLE.E.S. **Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real**. Disponível em: <[http://escalonamento-str.googlecode.com/hg-history/df07ebb92b94e78e5a7a8b29d4109ff51dd5b104/etc/JAI2009\\_Completo\\_Revisado.pdf](http://escalonamento-str.googlecode.com/hg-history/df07ebb92b94e78e5a7a8b29d4109ff51dd5b104/etc/JAI2009_Completo_Revisado.pdf)>. Acessado em :20 jan.2013.

PROCÓPIO, Felipe Victor; VERÍSSIMO, Júlio Cesar & MIRANDA, Juliano Coelho. **Desenvolvimento de Dirigível Indoor controlado via Rádio Frequência**. Artigo de final de curso. UNIS- MG, Novembro, 2010. Belo Horizonte, Minas Gerais, Brasil. Disponível em:<[http://www.cienciadacomputacao.unis.edu.br/files/.../Artigo\\_Dirigível.pdf](http://www.cienciadacomputacao.unis.edu.br/files/.../Artigo_Dirigível.pdf)> .Acesso em: 09 dez.2012.

SALGADO, R. M. A. **Transmissão Wireless de Vídeo DVD/HD**. 2008. 62f. Dissertação (Mestrado em Engenharia Eletrotécnica) - Faculdade de Engenharia da Universidade do Porto, Faculdade do Porto, 2008.

SBEGHEN, R. C. **Processamento Digital de Imagem**. 2007. 53f. Monografia (Bacharelado em Ciência da Computação ) - Faculdade de Jaguariúna, Jaguariúna , 2007. Disponível em: <<http://bibdig.poliseducacional.com.br/document/?code=51>> Acesso em: 10 jan. 2013

SILVA, A. M. E. **Monitoramento de Ambiente Através de uma Unidade Móvel Controlada pelo Computador**. 2010. 111f. Monografia (Bacharelado em Engenharia De Computação)CentroUniversitárioDebrasília,Brasília,2010.Disponívelem:<[http://repositorio.uniceub.br/bitstream/1\\_23456789/980/1/9965556.pdf](http://repositorio.uniceub.br/bitstream/1_23456789/980/1/9965556.pdf)>. Acesso em: 29 nov. 2012.

ZACCARIOTTO, V. L. **Deteccção De Pessoas Utilizando Vídeo em Ambiente Controlado**. 2010. 49f. Monografia (Bacharelado em Ciência da Computação) - Universidade São Francisco, Itatiba, São Paulo, 2010. Disponível em: <<http://people-detection-csharp.googlecode.com/files/Paper.pdf>> .Acesso em: 18 dez .2012.

VIEIRA, A. M. **Desenvolvimento De Um Módulo De Descrição De Imagens Para Um Sistema De Recuperação Em Um Banco De Imagens**. 2008. 71f. Monografia (Bacharelado em Ciência da Computação ) - Faculdade De Informática De Presidente PRUDENTE,2008.Disponívelem:<[http://fipp.unoeste.br/~chico/FIPP/projetos/projeto2008/Monografia\\_Anderson\\_Vieira\\_2008.pdf](http://fipp.unoeste.br/~chico/FIPP/projetos/projeto2008/Monografia_Anderson_Vieira_2008.pdf)>. Acesso em: 17 jan. 2013



**APÊNDICE I**

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Windows.Forms;
7 using System.Linq;
8 using System.Text;
9 using AForge.Video; // biblioteca de processamento
10 using AForge.Video.DirectShow; // biblioteca de video
11 using System.Drawing.Imaging;
12 using System.IO.Ports;
13 using System.Data.SqlClient; //biblioteca para Banco de Dados
14
15 using System.IO; // biblioteca das Portas
16
17 namespace TCC2
18 {
19     public partial class Form1 : Form
20     {
21         public Form1()
22         {
23             InitializeComponent();
24             BuscarDispositivos();// função busca os dispositivo conectado
25
26         }
27
28         private Boolean com; // variavel para Status
29         private string[] portas; // variavel para lista as portas
30         private bool TemDispositivos = false;
31         private FilterInfoCollection DispositivosDeVideos; // classe lista de dispositivos
```

USB

Directshow filtro de objetos

32 private VideoCaptureDevice EntradaVideo = null; // Esta classe permite ter acesso a

diferentes tipos de câmeras web USB ou outros dispositivos diferentes, o que suporta a interface

DirectShow..

33 private string conexaoBD = "Data Source=.\SQL;AttachDbFilename=C:\\Users\\lima\\Documents\\Visual Studio 2010\\Projects\\TCC2\\TCC2\\BDimagem.mdf;Integrated Security=True;Connect Timeout=30;

User Instance=True ";// local do banco

34 private byte[] vetorImagens;// bytes inteiro de 8 bits sem sinal.

35 private SqlConnection conexaoSQLServer; // variavel para fazer a conexão

36 SqlCommand Sql = default(SqlCommand);

37

38 ----- CONFIGURAÇÃO DA PORTAS COM SERIAL -----

private void controle\_Load(object sender, EventArgs e)

41 {

42

43 serialPort1.Close(); // metodo par abrir a porta

44 serialPort1.BaudRate = 9600; //taxa de tramitir

45 serialPort1.DataBits = 8; //

46 serialPort1.Parity = Parity.None;

47 serialPort1.StopBits = StopBits.One;

48 serialPort1.Handshake = Handshake.None;

49 serialPort1.Encoding = System.Text.Encoding.Default;

50

51 }

52

53 //-----LISTA AS PORTA DO COMPUTADOR-----

54

55

56 private void listaCOM() // funOo para lista a portas COM do PC

```

57 {
58 portas = SerialPort.GetPortNames(); //obter ma lista de nomes de porta serial.
59 if (portas.Length > 0)
60 {
61 PORTACOM.Items.Clear();
62 PORTACOM.SelectedIndex = -1;
63 foreach (string l in SerialPort.GetPortNames()) //Mostra o nome de cada porta
para o
console.foreach (porta corda nos portos)
64 {
65 PORTACOM.Items.Add(l); // LISTA A AS PORTA
2 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
66 }
67
68 }
69 else
70 {
71
72 MessageBox.Show("Camera não encontrada");
73 }
74
75
76
77 }
78 //-----botão iniciar-----
--79 private void iniciar_Click(object sender, EventArgs e)
80 {
81 if (PORTACOM.Text != "")// verificar a porta se esta aberta vazia faz outra coisa
82 {
83 try
84 {
85 //serialPort2.PortName = PORTACOM.Text;
86 // serialPort1.Open();
87 if (!serialPort1.IsOpen) // se a porta foi aberta ou não

```

```

88 serialPort1.PortName = PORTACOM.Text; // nomear as portas
89 serialPort1.Open();// abrir a porta COM
90 PORTACOM.Enabled = false;// desativa as porta
91 button4.Enabled = true;
92 ligar.Enabled = false;
93 esque.Enabled = true;
94 dire.Enabled = true;
95 toolStripStatusLabel2.Text = "Comunicação aberta da Porta COM";// ESTADO
96 }
97 catch
98 {
99     MessageBox.Show("Ocorreu um erro. Veja se a porta correta foi selecionada e
o
dispositivo,");
100 }
101
102 }
103
104 }
105 private void ligar_Click(object sender, EventArgs e)
106 {
107
108     listaCOM();
109     PORTACOM.Enabled = true;
110     iniciar.Enabled = true;
111 }
112
113 // -----BUSCA OS DISPOSITIVO QUE ESTOO LIGADO NO U
public void BuscaDispositivo(FilterInfoCollection Dispositivos) // função de
dispositvos do usb
115 { // Enumerar dispositivos de vídeo
116     for (int i = 0; i < Dispositivos.Count; i++)
117         proDispositivos.Items.Add(Dispositivos[i].Name.ToString());
118     proDispositivos.Text = proDispositivos.Items[0].ToString();

```

```
119 }
120
121 public void BuscarDispositivos() // função para buscar os dispositivos
122 {
123     DispositivosDeVideos = new
FilterInfoCollection(FilterCategory.VideoInputDevice);
124 if (DispositivosDeVideos.Count == 0)
125     TemDispositivos = false;
126 else
127 {
128     TemDispositivos = true;
129     BuscaDispositivo(DispositivosDeVideos);
130
131 }
132
133 }
134
135 public void TerminarVideo()// entrada de video
136 {
3 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
137 if (!(EntradaVideo == null))
138 if (EntradaVideo.IsRunning)
139 {
140     EntradaVideo.SignalToStop();// sinal para parar
141     EntradaVideo = null;
142 }
143 }
144
145 //
146 // PARA FAZER OS FRAMES DO VIDEOS E FEITO UM INSTANCIADA
IMAGEM
147
148 private void video_NovoFrame(object sender, NewFrameEventArgs eventArgs)
149 { //Manipulador de eventos novo quadro
```

```

150
151 Bitmap image = (Bitmap)eventArgs.Frame.Clone();
152 //Inicializa uma nova instancia da classe Bitmap a partir da imagem existente
    especificado.
153 pictureBox1.Image = image;
154 // Clone () Cria uma copia exata desta imagem recebida da camera . (Herdado
    de Imagem ).
155 }
156
157
158 //-----Captura a imagem para PICTUREBOx2 da camera-----
    -----159     private void captu2_Click(object sender,
    EventArgs e)
160 {
161     captu(); // chamar um metodo captu
162     //foto = true;
163 }
164 //-----Função para Captura Imagem-----
165 private void captu()//funao de captura a imagem da pictureBox1
166 {
167     try
168     {
169         if (EntradaVideo.IsRunning)
170         {
171             pictureBox2.Image = pictureBox1.Image;// salva a imagem e pictureBox
172         }
173     }
174
175     catch
176     {
177         MessageBox.Show("SEM IMAGEM PARA CAPTURA");
178
179     }
180 }

```

```

181
182 //-----Button PARA INICIAR A CAMERA PLAY-----
-----183 private void button1_Click(object sender, EventArgs e)
184 {
185 // estado.Text = "recebendo IMAGEM";
186
187 if (TemDispositivos)
188 {
189
190             EntradaVideo             =             new
VideoCaptureDevice(DispositivosDeVideos[proDispositivos.
SelectedIndex].MonikerString);
191 EntradaVideo.NewFrame += new NewFrameEventHandler(video_NovoFrame);
// criar video
source
192 EntradaVideo.Start();
193 proDispositivos.Enabled = false;
194             groupBox1.Text             =
DispositivosDeVideos[proDispositivos.SelectedIndex].Name.ToString();
195 Status(true);
196 }
197
198 else
199 {
200 try
201 {
202 if (EntradaVideo.IsRunning)
203 TerminarVideo();
204 proDispositivos.Enabled = true;
4 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
205 Status(false);
206 }
207 catch
208 {

```

```
209 MessageBox.Show("DISPOSITIVO NÃO ENCOMTRADO");
210 }
211 }
212 }
213
214 //-----estado da barra de menussss-----215 private void
timer1_Tick(object sender, EventArgs e)
216 {
217     tem.Text = DateTime.Now.ToLongTimeString();
218     tem.ForeColor = Color.Red;
219 }
220
221 private void sobreToolStripMenuItem_Click(object sender, EventArgs e)
222 {
223     //MessageBox.Show("PROGRAMA PARA RECEBE IMAGEM DE UM
CAMERA");
224
225
226 }
227
228 private void Status(bool teste)
229 {
230
231     if (teste == true)
232     {
233         camera.Text = "CONECTADO";
234         camera.ForeColor = Color.Green;
235         com = true;
236     }
237     else
238     {
239         camera.Text = "DESCONECTADO";
240         camera.ForeColor = Color.Red;
241         com = true;
```



```

242
243 }
244 }
245
246 ---Fecha porta Com-----248 private
void Fecha()
249 {
250 if (serialPort1 != null && serialPort1.IsOpen)
251 {
252 serialPort1.Close();
253 serialPort1 = null;
254 PORTACOM.Enabled = true;// desativa as porta
255 ligar.Enabled = true;
256 listaCOM();
257 }
258
259
260 }
261 private void button4_Click(object sender, EventArgs e)
262 {
263 Fecha();
264 }
265
266
267 -----268 //---- comandos para -----
-----269
270 protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
271 {
272 // verifica se a tecla esta pressionada
273 // é seta para esquerda
274 if (keyData == Keys.Left)
275 {
276 serialPort1.Write("4");//hexadecimal 26

```

5 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs

```
277 pictureBox4.Image = Properties.Resources.right_up;
278 pictureBox3.Image = Properties.Resources.esquerda;
279
280 }
281 else
282 if (keyData == Keys.Right)// direita
283 {
284 serialPort1.Write("6");//hexadecimal 26
285 pictureBox3.Image = Properties.Resources.left_up;
286 pictureBox4.Image = Properties.Resources.direita;
287
288 }
289
290 else
291 if (keyData == Keys.Down)
292 {
293 serialPort1.Write("8");//hexadecimal 26
294 pictureBox5.Image = Properties.Resources.bottom_up;
295 pictureBox6.Image = Properties.Resources.frente;
296
297 }
298 else
299 if (keyData == Keys.Up)
300 {
301 serialPort1.Write("2");//hexadecimal 26
302 pictureBox6.Image = Properties.Resources.top_up;
303 pictureBox5.Image = Properties.Resources.re;
304
305 }
306 else
307 if (keyData == Keys.Shift) // Tecla CTRL
308 {
309 serialPort1.Write("0");
310
```

```

311 }
312
313 return base.ProcessCmdKey(ref msg, keyData);
314 }
315
316
317 private void esque_Click(object sender, EventArgs e)
318 {
319
320 serialPort1.Write("1");//hexdecimal 26
321 }
322
323 private void dire_Click(object sender, EventArgs e)
324 {
325 serialPort1.Write("3");//hexdecimal 26
326 }
327
328 -----329 //-----button de stop-----
330 private void button2_Click(object sender, EventArgs e)
331 {
332 TerminarVideo();
333 proDispositivos.Enabled = true;
334 }
335
336
337
338 //-----Banco de dados-----
339
340 //PARA ATIVAR O BANCO SQL SERVE
341 private void btnAtivar_Click(object sender, EventArgs e)
342 {
343
344 //cria uma instância do objeto SqlConnection
345 conexaoSQLServer = new SqlConnection(conexaoBD);//passando como

```

parametro a string e

conexão ao banco

```
346 //obtem os dados da tabela imagens
```

```
6 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
```

```
347 ImagensSQLServer(conexaoSQLServer);
```

```
348 //ativa os botões de comando
```

```
349 //btnCarregarImagem.Enabled = true;
```

```
350 RetomarImagem.Enabled = true;
```

```
351 SalvaImagem.Enabled = true;
```

```
352 Exclu.Enabled = true;
```

```
353 btnAtivar.Enabled = false;
```

```
354 }
```

```
355
```

```
356 private void ImagensSQLServer(SqlConnection conexaoSQLServer) // Carrega
```

```
dados
```

```
357 {
```

```
358 try
```

```
359 {
```

```
360 // Inicializar o SQL adaptador
```

```
361 SqlDataAdapter BANCO = new SqlDataAdapter("Select Id,descricao,imagem  
from Imagens",
```

```
conexaoSQLServer);
```

```
362
```

```
363 //Inicializa o Dataset.
```

```
364 DataSet DATAS = new DataSet();
```

```
365
```

```
366 // Preenche o conjunto de dados com uma IMAGENS Tabela
```

```
367 BANCO.Fill(DATAS, "Imagens");// preeche todas as informações do DataSet
```

```
368
```

```
369 // preencher o datagridview com o conjunto de dados.
```

```
370 gridViewImagens.DataSource = DATAS.Tables["Imagens"];
```

```
371
```

```
372 }
```

```
373 catch (Exception ex)
```

```

374 {
375     MessageBox.Show(ex.ToString());
376
377 }
378 }
379 //-----onde vai fica as imagem do banco id-----
-
380 private void gdvImagens_CellClick(object sender, DataGridViewCellEventArgs
e)
381 {
382     //obtem o codigo da imagem e exhibe no controle textbox
383         txtCodigoImagem.Text =
(gdvImagens.Rows[e.RowIndex].Cells["id"].Value).ToString();//obtem o
código do cliente de uma linha selecionada no datagridview
384     txtDescricaoImagem.Text = "";
385
386 }
387
388 //PARA SALVA A IMAGEM NA BANCO DE DADOS -----
-----389
390 private void SalvaImagem_Click(object sender, EventArgs e)
391 {
392     int confirmar = 1;// variavel
393
394     if (string.IsNullOrEmpty(txtDescricaoImagem.Text))
395     {
396         //conexaoSQLServer.Open();
397         MessageBox.Show("Informe a descrição da imagem", "Código da Imagem",
MessageBoxButtons.
OK, MessageBoxIcon.Information);
398     return;
399
400 }
401

```

```

402 try
403 {
404 this.conexaoSQLServer.Open();//abri a conexão do banco
405 Byte[] image = new Byte[256];// salva a matriz de Bytes para banco de dados
406 image = Codifica(pictureBox2.Image); //busca a função Codifica e tranformar o
imagem em
bytes
407
408 this.Sql = new SqlCommand();
409 Sql.Connection = conexaoSQLServer;// fazer a conexão
410 if (Sql.Parameters.Count == 0)// condição
411 { // no COmmandText so SqlCommad,passo aintrução SQL referencia a inserção
dos dados
412     this.Sql.CommandText = "INSERT INTO Imagens(descricao,imagem)
values(@descricao,@
imagem)";// no COmmandText so SqlCommad,passo aintrução SQL referencia a
inserção dos dados
413     this.Sql.Parameters.Add("@descricao", System.Data.SqlDbType.VarChar,
150);
414 this.Sql.Parameters.Add("@imagem", System.Data.SqlDbType.Image);
415 }
7 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
416
417     this.Sql.Parameters["@descricao"].Value = this.txtDescricaoImagem.Text;
//adicionano o
valor textbox nos paramentos do comando
418 this.Sql.Parameters["@imagem"].Value = image; //this.vetorImagens;
419
420 txtDescricaoImagem.Clear();
421 txtCodigoImagem.Clear();//metodo para limpa da tela
422 int irestado = this.Sql.ExecuteNonQuery();//metodo par aabri a Conexão
423
424 // condição se banco na estiver ligado
425 if (irestado <= 0)

```

```

426 MessageBox.Show("Falha ao incluir imagem no banco de dados.");
427
428 ImagensSQLServer(conexaoSQLServer);
429
430 }
431 catch (Exception ex)
432 {
433     MessageBox.Show("Erro ao Salva");
434     throw ex ;
435     //conexaoSQLServer.Close();
436 }
437 finally
438 {
439     this.conexaoSQLServer.Close();// fecha a conexao
440     //conexaoSQLServer.Dispose();//objetivo de liberar os recursos de classe que
nã usado
441     confirmar = 1;
442 }
443
444 if (confirmar == 1)
445 {
446     MessageBox.Show(" Salvo Com Sucesso");
447     // Aqui aparece uma mensagem de confirmação caso ocorra algum erro, o
catch acima
exibe a mensagem dele
448 }
449
450
451 }
452 //-----RETONAR IMAGEM DO BANCO-----
-----453                                     private     void
RetomarImagem_Click(object sender, EventArgs e)
454 {
455

```

```
456 if (txtCodigoImagem.Text == string.Empty)
457 {
458     MessageBox.Show("Informe o Código da Imagem no Banco de dados",
"Código da Imagem",
MessageBoxButtons.OK, MessageBoxIcon.Information);
459 return;
460 }
461
462 try
463 {
464     if (txtCodigoImagem != null || !txtCodigoImagem.Equals(""))
465     {
466         SqlCommand cmdSelect = new SqlCommand("select imagem from Imagens
where id=@ID",
this.conexaoSQLServer);
467         cmdSelect.Parameters.Add("@ID", SqlDbType.Int, 4);
468         cmdSelect.Parameters["@ID"].Value = this.txtCodigoImagem.Text;
469         this.conexaoSQLServer.Open();// abrir a conexão
470
471         byte[] vetorImagem = (byte[])cmdSelect.ExecuteScalar();// retorna a primeira
linha e
primeira coluna do comando
472
473         string NomeArquivo = Convert.ToString(DateTime.Now.ToFileTime());
474         FileStream fs = new FileStream(NomeArquivo, FileMode.CreateNew,
FileAccess.Write);
475         fs.Write(vetorImagem, 0, vetorImagem.Length);
476         fs.Flush();
477         fs.Close();
478         // converter a matriz de bytes de imagem para imagem Real
479
480         pictureBox2.Image = Image.FromFile(NomeArquivo);// onde esta a imagem
para ser
capturada
```



```
481 }
482
8 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
483 else
484 {
485     MessageBox.Show("Registro excluido");
486 }
487
488 }
489 catch
490 {
491     MessageBox.Show("Dados Invalidos");
492 }
493
494 finally
495 {
496     this.conexaoSQLServer.Close();
497     // conexaoSQLServer.Dispose();//objetivo de liberar os recursos de classe
498 }
499 }
500
501 // para sair da aplicação
502 private void sairToolStripMenuItem_Click(object sender, EventArgs e)
503 {
504     Application.Exit();
505 }
506
507 //-----Codifica a imagem-----
-----508
509 public static Byte[] Codifica(Image foto)
510 { // contem os bytes da imagem PictureBox
511     MemoryStream memoria = new MemoryStream();
512     // classe MemoryStream ler dados no fluxo de memória,para converte para
dados bytes
```

```

513 foto.Save(memoria, ImageFormat.Jpeg);// save imagem em formato Jpeg
514 Byte[] MyData = new Byte[memoria.Length]; // transferencia de bytes o array
515 // Converte a entrada em saída bytes
516 MyData = memoria.ToArray();
517 // Escreve a segunda seqüência para o fluxo, byte do array.
518 return MyData;
519 }
520
521 //-----
-----
522
523 //-----DELETE-----
-----524
525 private void Exclu_Click(object sender, EventArgs e)
526 {
527
528 if (txtCodigoImagem.Text == string.Empty)
529 {
530 MessageBox.Show("Informe o Código para ser Excluido", "Código da Imagem",
531 MessageBoxButtons.OK, MessageBoxIcon.Information);
532 return;
533 }
534 else
535 try // tenta executar o que estiver abaixo
536 {
537 SqlCommand cmdSelect = new SqlCommand("delete from Imagens where id =
538 @ID",this.
539 conexaoSQLServer);
540 cmdSelect.Parameters.Add("@ID", SqlDbType.Int, 4,
541 "@ID");//(gdvImagens.Rows[e.RowIndex].
542 Cells["id"].Value).ToString());
543 cmdSelect.Parameters["@ID"].Value = this.txtCodigoImagem.Text;
544 this.conexaoSQLServer.Open();

```

```
541 cmdSelect.ExecuteNonQuery();//executa cmdSelect
542
543 // pictureBox2.Image = Image.FromFile(strNomeArquivo);// onde esta a
imagem para ser
capturada
544 MessageBox.Show("Dados Excluidos ", "Mensagem", MessageBoxButtons.OK,
MessageBoxIcon.
Information);
545 }
546 catch (Exception ex)
547 {
548 //throw;
9 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
549 MessageBox.Show(ex.Message);
550 }
551
552 finally
553 {
554 this.conexaoSQLServer.Close();
555
556 ImagensSQLServer(conexaoSQLServer);
557
558 }
559 }
560
561 private void button3_Click(object sender, EventArgs e)
562 {
563 // gridView1.Clear(); //Para a grid se atualizar
564 DialogResult resultado = MessageBox.Show("Confirma exclusão da Foto",
"confirmar",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
565 if (resultado == DialogResult.Yes)
566 {
567 try // tenta executar o que estiver abaixo
```

```
568 {
569
570
571 SqlCommand cmdSelect = new SqlCommand("delete from Imagens where id =
@ID", this.
conexaoSQLServer);
572 cmdSelect.CommandType = CommandType.StoredProcedure;
573             cmdSelect.Parameters.Add("@ID",             SqlDbType.Int,
4);//(gdvImagens.Rows[e.RowIndex].
Cells["id"].Value).ToString());
574 cmdSelect.Parameters["@ID"].Value = this.txtCodigoImagem.Text;
575
576
577 this.conexaoSQLServer.Open();
578 cmdSelect.ExecuteNonQuery();//executa cmdSelect
579
580 // pictureBox2.Image = Image.FromFile(strNomeArquivo);// onde esta a
imagem para ser
capturada
581 MessageBox.Show("Dados Excluidos ", "Mensagem", MessageBoxButtons.OK,
MessageBoxIcon
.Information);
582 }
583 catch (Exception ex)
584 {
585 //throw;
586 MessageBox.Show(ex.Message);
587 }
588
589 finally
590 {
591 this.conexaoSQLServer.Close();
592
593 ImagensSQLServer(conexaoSQLServer);
```

```
594 }
595
596 }
597 else
598 {

599 if (resultado == DialogResult.No)
600     MessageBox.Show("Numero não exitir", "Mensagem",
        MessageBoxButtons.OK, MessageBoxIcon.
        Information);
601 }
602

603 }
604 //-----busca as outras janelas-----
605 private void aplica_Click(object sender, EventArgs e)
606 {
607     Form2 newForm2 = new Form2();
608     newForm2.ShowDialog();
609 }
610

611 private void ajuda_Click(object sender, EventArgs e)
612 {
613     Form3 newForm3 = new Form3();
614     newForm3.ShowDialog();
615 }
616

10 C:\Users\lima\Desktop\TCC2\TCC2\Form1.cs
617

618 }//chave do form
```

```
619 }// final
```

```
620
```