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

RAFAEL DE FARIAS MEURER
VINICIUS DA SILVA ARCANJO

SISTEMA DE MONITORAMENTO GUIADO REMOTAMENTE

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

RAFAEL DE FARIAS MEURER
VINICIUS DA SILVA ARCANJO

SISTEMA DE MONITORAMENTO GUIADO REMOTAMENTE

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Eletrônica e Departamento Acadêmico de Informática Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Engenheiro em Computação".

Orientador: Prof. Dr. Carlos Raimundo Erig Lima

CURITIBA

2014

AGRADECIMENTOS

Gostariamos de agradecer nosso colega de classe Fernando Padilha, pelo empréstimo de diversos itens os quais efetivamente colaboraram para a construção deste projeto. Além disso, agradecemos também ao Robert C. Nelson, por disponibilizar o sistema operacional Linux Ubuntu para a placa beaglebone. Por fim, gostariamos de agradecer todos que estão relacionados neste projeto de alguma forma.

RESUMO

MEURER, Rafael de Farias e ARCANJO, Vinicius da Silva. Sistema de Monitoramento Guiado Remotamente. 56 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Eletrônica e Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Devido o aumento da sensação de insegurança, muitas pessoas estão optando por contratar serviços de segurança privada. Dentre estes, o mais procurado é a vigilância por câmeras, esta tecnologia passou por diversas mudanças até chegar no estado atual, onde é possível ter câmeras acessadas via IP. A próxima etapa é fazer com que as câmeras tenham liberdade de locomoção e rotação, para aumentar a área de abrangência de monitoramento. Com isto em mente, este projeto consiste em construir um sistema de monitoramento guiado remotamente. Para a construção do mesmo, foi desenvolvido um robô, onde a câmera fica acoplada, e uma interface gráfica, programada em Java, para que o usuário possa conectar-se e controlar o robô, e monitorar o ambiente. O projeto utiliza-se de uma interface de rede *wireless* 802.11, para uma mobilidade maior do robô, um sistema operacional Linux embarcado, através do qual é possível utilizar todos os *drivers* dos periféricos, além de contar com a pilha TCP/IP implementada, a qual é essencial no projeto devido a necessidade de utilização dos protocolos de transporte UDP e TCP. O *software* de controle do sistema, que roda no Linux Embarcado, foi programado em Python, este *software* recebe comandos da interface gráfica e os executa. A câmera utilizada é uma *web-cam*, a qual possui resolução satisfatória para o monitoramento de vídeo, esta possui graus de liberdade mesmo com o robô parado. O vídeo do ambiente pode ser comprimido através do algoritmo MPEG-4 parte 2, utilizando a biblioteca GSTreamer, caso o usuário deseje diminuir o consumo de banda. Ao termino do projeto, foram realizados testes para verificação dos requisitos do sistema, os quais apresentaram sucesso, consequentemente o objetivo geral deste projeto foi atingido.

Palavras-chave: robô, monitoramento, remoto, guiado

ABSTRACT

MEURER, Rafael de Farias e ARCANJO, Vinicius da Silva. Guided Remotely Monitoring System. 56 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Eletrônica e Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Due to the increasing sense of insecurity, many people are choosing to hire private security services. Among these, the most wanted is the surveillance cameras, this technology has been changed in many ways until it gets at the current state, which can be accessed via IP cameras. The next step is to get the cameras the freedom of movement and rotation in order to increase the coverage area of monitoring. With this point in mind, this project aims to design a monitoring system remotely guided. For its construction, a robot was designed, where the camera is attached, and a graphical user interface, programmed in Java, so the user can connect and control the robot, and watch the environment. This project makes use of a network interface 802.11 wireless for greater mobility robot, a Linux embedded operating system, which allows you to use all the peripherals drivers, in addition to having TCP/IP stack implemented, which is essential in this project due to the need of using UDP and TCP transporting protocols. The control software of the system, which runs on a Embedded Linux operating system, was coded in Python, this software receives commands from the graphical interface and executes them. The camera used is a web-cam, which has got satisfactory resolution for monitoring, this camera has degrees of freedom even with the robot stopped. The video of the environment can be compressed using the MPEG-4 Part 2 algorithm, by using the GStreamer library, if the user wants to reduce the bandwidth consumption. At the end of the project, tests to check the system requirement were done, which were successful, therefore the goal of this project was successfully achieved.

Keywords: robot, monitoring, remote, guided

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ABS	*Antilock Breaking System*
ARM	*Advanced RISC Machine*
codec	*compression decompression*
DCT	*Discrete Cosine Tranforme Cosine*
DDR2	*Double Data Rate 2*
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name Server*
DVR	*Digital Video Recorder*
FPS	*Frames Per Second*
IMC	Interface de Monitoramento e Comandos
IPEA	Instituto de Pesquisa Econômica
IP	*Internet Protocol*
ISM	*Industrial, Scientific and Medical*
JNA	*Java Native Access*
LEDs	*Light Emissor Diodes*
LGPL	*Lesser General Public License*
LI-PO	*lithium polymer*
MPEG-4	*Moving Picture Expert Group*
PCB	*Printed Circuit Board*
PWM	*Pulse Width Modulation*
QoS	*Quality of Service*
RAM	*Random Access Memory*
SE	Sistema Embarcado
SFE	Segurança Física Eletrônica
SIA	*Security Industry Association*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
USB	*Universal Serial bus*
VCR	*Video Camera Recorder*
VoIP	*Voice over Internet Protocol*
WLAN	*Wireless Local Area Network*
WPAN	*Wireless Personal Area Network*

SUMÁRIO

1 INTRODUÇÃO

No Brasil a sensação de insegurança faz parte do cotidiano da população, conforme a pesquisa realizada pelo IPEA sobre segurança pública, 61,6% dos entrevistados têm muito medo de terem suas residências arrombadas e 62,4% de serem assaltadas a mão à armada. Além deste medo, une-se a baixa confiança mostrada sobre as instituições policiais (JUNIOR; ALENCAR, 2012). Devido as estes fatos, a população tem recorrido a soluções que garantam uma maior segurança, esta procura aumenta a demanda sobre as soluções de Segurança Física Eletrônica (SFE).

O resumo executivo apresentado pela *Security Industry Association* Brasil (SIA Brasil) em 2012, demonstra um aumento de 18,5% nas receitas das empresas de SFE, entre 2011 e 2012. Além deste aumento, a projeção para os anos seguintes neste setor são ainda melhores, a figura 1 apresenta um gráfico sobre este crescimento até 2017. No mercado de SFE existem diversas soluções oferecidas, a mais utilizada é a vigilância por câmera, 39,6% do mercado. Este tipo de vigilância, consiste no monitoramento através de câmeras em um determinado ambiente (KEMPER, 2012).

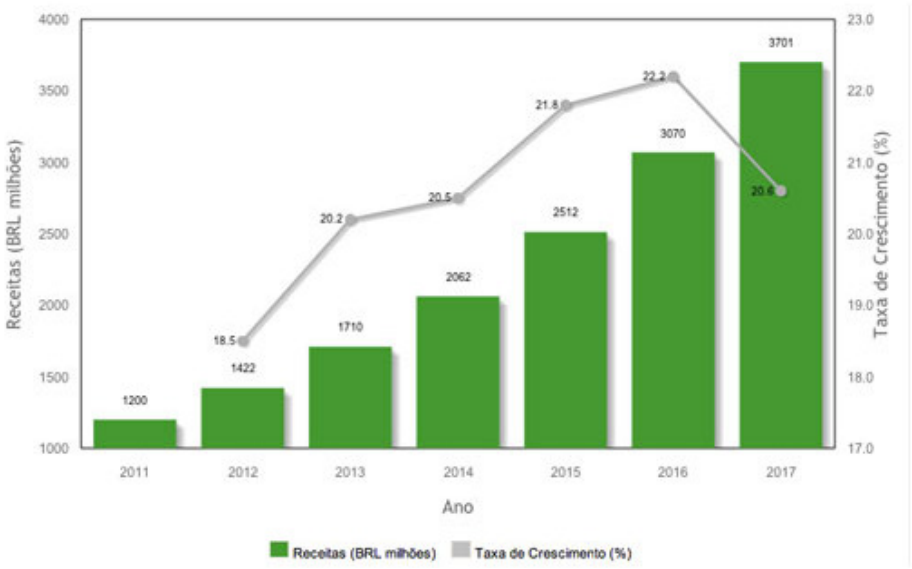

Figura 1: Receita e Taxa de Crescimento prevista para os próximos anos da Security Industry Association Brasil (KEMPER, 2012)

Inicialmente o ambiente era filmado por câmera analógicas e transmitida através de monitores, o vídeo era armazenado através de fitas pelo VCR. O alto custo de manutenção e a troca frequente das fitas levou ao desenvolvimento de uma nova solução, o DVR, onde o armazenamento é realizado por mídias digitais. Ambas tecnologias apresentam a limitação de que a estação de monitoramento é fixa. Com a necessidade de realizar a monitoração em tempo real através de outras estações, o transporte de vídeo passou a ser realizado através do protocolo IP. Isto faz com que qualquer estação, que possua acessibilidade IP, possa monitorar o ambiente (FULLERTON; SALES, 2008).

Com o intuito de obter um maior detalhamento do ambiente, câmeras acopladas a robôs remotamente guiados começaram a surgir. Desta maneira, a pessoa responsável pelo monitoramento possui a liberdade de deslocar o robô para determinados locais, onde previamente não seria possível realizar o monitoramento.

1.1 OBJETIVO GERAL E OBJETIVOS ESPECÍFICOS

Esse trabalho tem como objetivo geral desenvolver um sistema remoto de monitoramento guiado. Composto por uma câmera acoplada a um robô e uma interface remota de monitoramento onde, através desta o usuário poderá monitorar as imagens capturadas pela câmera, conforme demonstra a figura 2. Este objetivo geral será dividido nos seguintes objetivos específicos:

- Modelar o sistema como um todo;
- Projetar e implementar o sistema de comunicação sem fio, para enviar e receber dados e vídeo;
- Implementar o sistema embarcado, onde ficará o robô com a câmera;
- Implementar a interface remota do usuário, para que possa controlar o robô e monitorar o ambiente;

Figura 2: Visão do projeto (Autoria Própria)

1.2 ORGANIZAÇÃO DO DOCUMENTO

Esse documento será organizado da seguinte forma. No capítulo 2 será apresentado a fundamentação teórica e trabalhos correlatos. A metodologia utilizada, as tecnologias e fundamentos para desenvolver o projeto no capítulo 3. No capítulo 4 será demonstrado o desenvolvimento do projeto, e o resultado final. Será apresentado um comparativo das horas e custos planejados e o real no capítulo 5. Por fim, o capítulo 6 apresenta a conclusão do projeto.

2 REFERENCIAL TEÓRICO

Neste capítulo será apresentado os trabalhos correlatos. Posteriormente, será apresentado uma abordagem geral sobre as tecnologias que estão relacionadas ao projeto, e quais tecnologias e fundamentos foram utilizados no projeto.

2.1 TRABALHOS CORRELATOS

Dentre os trabalhos encontrados existem tanto trabalhos acadêmicos quanto comerciais. Os trabalhos acadêmicos mostram o que está sendo desenvolvido como pesquisa atualmente, enquanto os comerciais mostram os produtos existentes no mercado. Estes trabalhos correlatos serão comparados com o projeto que propomos para verificar se o projeto é relevante para o meio acadêmico e também se é financeiramente viável para se tornar um produto.

Dentre os trabalhos acadêmicos pesquisados, um que mostrou-se relevante ao projeto foi o desenvolvimento de um robô controlado através da Internet por um usuário, utilizando TCP/IP, que acessa uma interface de controle em um servidor *web*. Neste trabalho, o autor cita a importância de existir uma interface que contenha apenas informações relevantes ao usuário. A arquitetura da comunicação final do usuário com o robô foi dividida em três camadas, a primeira situa o cliente enviando comandos ao robô, a segunda o robô recebendo os comandos do cliente e a última camada apresenta as imagens e valores de sensores coletados do ambiente para o robô são enviados para o cliente (SUGISAKA; HAZRY, 2005).

Um outro projeto, relacionado ao que estamos propondo foi analisado, pois nele o autor tem como objetivo fazer um robô que consiga comunicar sem fio com um servidor e o que um usuário final possa acessar via Internet o robô, e receber as imagens capturadas da câmera. O artigo apresenta rapidamente dois protocolos de comunicação sem fio, e cita que irá usar o padrão 802.11b/g, pelo grande número de pontos de acesso e também pela taxa de transferência de dados que este padrão pode proporcionar para transmitir vídeos. O programa para que usuário final possa enviar e receber comandos, foi escrito em Java applet porque, de

acordo com o artigo, os números de equipamentos que possam executar o programa, e assim controlar o robô, é maior. Além disso, o autor cita que ele teve problemas com a taxa de transmissão da imagem da câmera com o microcontrolador, para solucionar o problema foi necessário fazer uma modificação na taxa de transmissão da câmera (GUPTA et al., 2008).

Em relação aos produtos comerciais correlatos, o primeiro produto é de um robô que possui uma interface 802.11b/g e pode ser acessado de qualquer computador que possua acesso a Internet e um *web browser*. Este robô captura imagem e som do ambiente através de uma câmera que pode ser controlada, juntamente com um microfone. Além disso, rotas podem ser definidas para que o robô possa se locomover de forma autônoma (WOWWEE.COM, 2013). O preço deste robô é US$ 299 (AMAZON.COM, 2013a).

Em relação ao segundo robô comercial (SPYKEEWORLD.COM, 2013), também tem os mesmos objetivos, com a adição de novas funcionalidades como, tocador de música, utilização de VoIP para fazer chamadas e tirar fotos na parte de monitoramento. O robô contém um sensor de detecção de movimentos e caso encontre algo, o robô irá enviar um *email* para o dono do robô com a imagem do intruso. Existem duas de formas de acesso ao robô, uma local via 802.11, outra remota, utilizando a internet, porém necessário um ponto de acesso à um *gateway* disponível. O preço encontrado foi de US$ 444,52 (AMAZON.COM, 2013b).

Fazendo uma comparação com com o robô que estamos desenvolvendo com o primeiro robô comercial (WOWWEE.COM, 2013), a única característica distinta que não irá ter no robô é a questão de se locomover de forma autônoma através de rotas definidas pelo usuário, em relação ao resto das características são praticamente todas similares, eventualmente pode haver algumas diferenças tal como resolução da qualidade da câmera, do microfone e protocolos que estarão envolvidos nos transporte de vídeos e dados. Entretanto, de uma forma geral podemos dizer que o nosso projeto se assemelha muito a este produto. Em relação ao segundo (SPYKEEWORLD.COM, 2013), pode se dizer que o projeto também é semelhante, mudando poucas coisas principalmente no fato de ter um realizar chamadas VoIP e tocador de música, tais facilidades no projeto, iria acrescentar uma despesa maior em relação ao *hardware*, e provavelmente a relação custo benefício para o cliente no produto final não seria vantajosa.

2.2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentada a fundamentação teórica sobre processamento de imagens e compressão de vídeo, redes sem fio e sistemas embarcados.

2.2.1 PROCESSAMENTO DE IMAGENS E COMPRESSÃO DE VÍDEO

Quando um conjunto sequencial de imagens são exibidas através de transições rápidas um fenômeno conhecido como persitência da visão ocorre. Ao invés de perceber cada imagem separadamente, um efeito de movimentação contínua é observado. Este fenômeno ocorre quando a frequência de imagens por segundo (FPS) é superior a 16 FPS. Nesta frequência, o efeito *flicker*, que se caracteriza por uma percepção visual de não continuidade entre as trocas de *frames*, não estará presente. Embora esta percepção seja uma questão subjetiva, muitos acreditam que é necessário um valor entre 24 a 30 FPS, para o efeito *flicker* seja completamente removido (BEACH, 2008).

Segundo as autoras Petrou e Bosdogiani (1999), o processamento de imagens tem sido desenvolvido em resposta a diversos problemas, sendo um destes, a necessidade de realizar a codificação de imagens para que a transmissão destas seja facilitada. A função primária da compressão de vídeo é definida por realizar uma codificação, a qual irá reduzir a quantidade de bits necessária para realizar o armazenamento ou a transmissão de vídeo. Esta codificação do formato de entrega do vídeo deverá ser realizada dependendo do meio envolvido, por exemplo, TV, DVD, Internet, dentre vários outros. A dificuldade da realização deste processo, encontra-se no fato de que as especificações técnicas e as limitações do meio deverão ser respeitadas e o resultado da codificação deve produzir uma experiência audiovisual que seja satisfatória para o usuário final (BEACH, 2008).

É importante ressaltar que, segundo o autor Sayood (2006), existem dois tipos de compressão, denominados *lossy* e *lossless*. A diferença entre estes tipos, é que o primeiro provoca perdas de representação quando a decompressão for realizada. Devido a esta característica, o tipo *lossy* consegue realizar uma compressão mais eficaz. Entretanto, nem sempre a compressão que possui uma maior eficácia é a mais adequada tendo em vista que, dependendo da aplicação perdas de representação não são toleráveis. Portanto, para cada aplicação deve ser realizada uma análise do *tradeoff* de compressão em relação a necessidade de representação fiel do padrão original.

De acordo com o autor Beach (2008), a compressão de vídeo é feita através da análise do conteúdo de cada *frame*, com o objetivo de encontrar uma maneira de como representar este conteúdo utilizando menos *bits*. Este feito é obtido por algoritmos de codec que tratam este problema de diversas formas. De uma forma geral, o que um algoritmo de codec faz é detectar regiões denominadas *macroblocks*, que por definição são regiões na imagem que possuem uma determinada variação de cores e brilho comum entre si de acordo com um determinado *threshold*. Posteriormente, ocorre uma representação destes *macroblocks* com determinados

números, através da Transformada Discreta de Coseno - DCT, que possibilitam a reconstrução do padrão original.

Um estudo comparativo entre diversos algoritmos de codecs utilizados por determinadas aplicações, realizado por Golston (2004), mostra que, de uma forma geral, considerando os *tradeoffs* envolvidos, diversas aplicações de monitoramento de segurança utilizam amplamente o codec MPEG-4, o qual oferece um bom balanço entre eficácia de compressão, qualidade da imagem e utilização de recursos do processador. Além disso, o autor comenta que, em casos que foram necessário obter uma maior eficácia de compressão foi utilizado o codec H.264. A figura 3 e 4 ilustra, respectivamente, a codificação e decodificação realizada pelo algoritmo H.264, também conhecido como MPEG-4 parte 10.

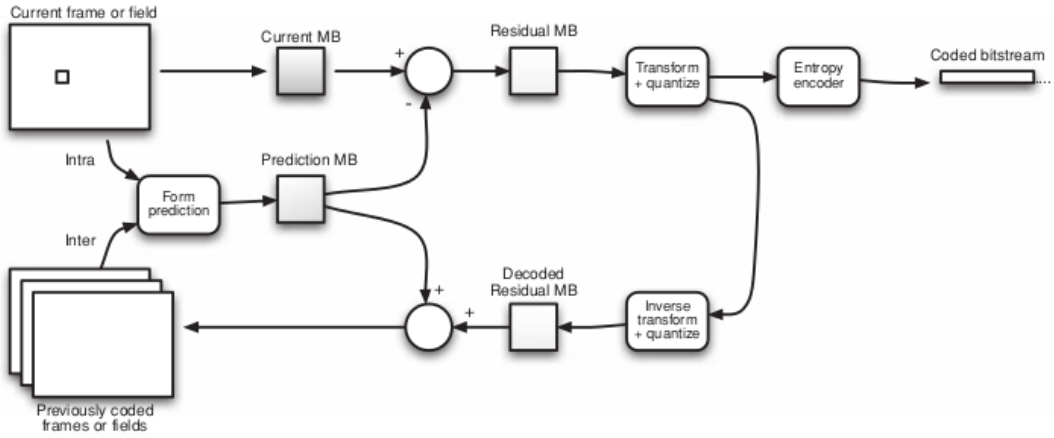

Figura 3: Codificação H.264 (RICHARDSON, 2010)

Para processo de codificação do vídeo é necessário que a cada *macroblock* criado, formado por uma matriz de 16x16 *pixels*, exista uma previsão do sucessor. Subtraindo estes dois *macroblock*, obtém-se um *macroblock* residual, neste, é realizado uma DCT. A saída desta transformada é um conjunto de coeficientes, o qual será quantizado dividindo-se por um número inteiro. Ao final desta operação, é obtida a codificação do vídeo, denominada *coded bitstream*. Além disso, o resultado da transformada juntamente com a quantização é utilizado como uma realimentação, aplicando-se a DCT inversa e uma nova quantização. Este resultado será somado com o *macroblock* que foi previsto, para que novas previsões futuras sejam realizadas (RICHARDSON, 2010).

Para decodificação do vídeo, um ajuste de escala e a DCT inversa é aplicado no *coded bitstream* recebido, resultando em um *macroblock* residual. Posteriormente, realiza-se a mesma previsão de *macroblock* utilizada pelo codificador. Esta previsão é adicionada ao *macroblock* residual e o resultado final é obtido (RICHARDSON, 2010).

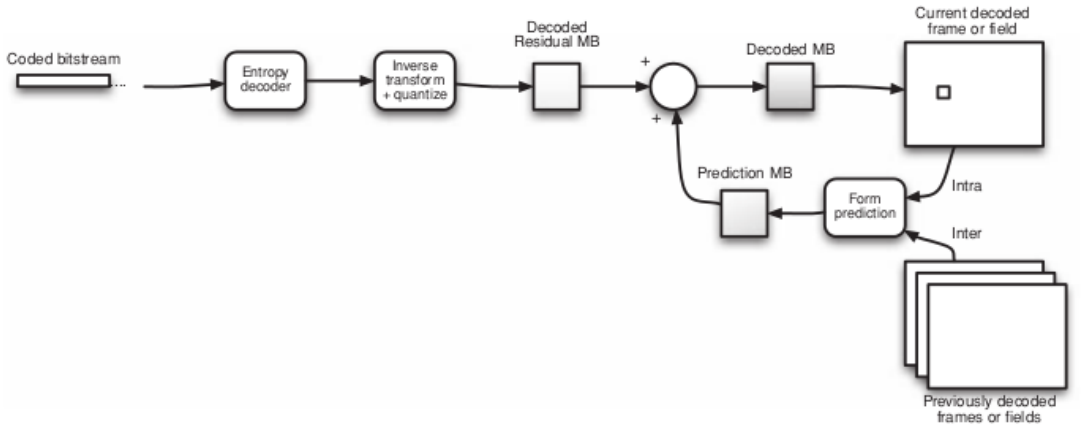

Figura 4: Decodificação H.264 (RICHARDSON, 2010)

O outro algoritmo analisado é o MPEG-4 parte 2, a diferença de funcionamento entre este e o H.264 está nos parâmetros de configuração dentre eles, e alguns métodos selecionados para transformação de imagem. O quadro 1 apresenta as principais diferenças entre os dois algoritmos.

Quadro 1: Principais diferenças entre MPEG-4 e H.264 Adaptado de (RICHARDSON, 2003)

Comparação	**MPEG-4**	**H2.64**
Eficiência de Compreensão	Média	Alta
Transformada	8x8 DCT	4x4 DCT aproximada
Bloco Mínimo para compreensão	8x8	4x4
Suporte para Vídeo Streaming	Codificação Escalável	Alteração de fatias
Precisão do Vetor de Movimento	metade ou um quarto de pixel	um quarto de pixel

2.2.2 REDES SEM FIO

Segundo o autor, Papadimitriou et al. (2002), a área de redes *wireless* tem apresentado um enorme crescimento, tornado-se um dos setores mais importantes da indústria de telecomunicações. Devido a esta ascensão e a importância dos sistemas de comunicação *wireless* no dia a dia, o autor cita que existe a expectativa que irá chegar um momento onde a quantidade de usuários que utilizam dispositivos *wireless* será maior que a quantidade que utilizam dispositivos *wired*. Portanto, existe a expectativa de um contínuo crescimento destas redes.

De acordo com o autor, Papadimitriou et al. (2002), a ampla popularidade e crescimento das redes de comunicação *wireless* é devido as vantagens que esta tecnologia proporciona em relação as redes *wired*. A mais importante destas vantagens, é o fato de permitir

ao usuário a liberdade de não utilizar cabos. Desta maneira, possibilita um tipo de conexão que proporciona uma excelente mobilidade, a qual é desejada por diversas aplicações. Atrelado a este fator de mobilidade, as redes *wireless* possui a vantagem de poder disponibilizar o acesso a rede em determinados ambiente cujo o cabeamento físico não seja viável devido a limitações ou restrições existentes (PAPADIMITRIOU et al., 2002).

Devido a livre utilização da banda ISM (902-928 MHz, 2.4-2.5 GHz e 5.725-5.825 GHz), para transmissão de dados, existem diversos dispositivos que disputam a utilização destas frequências em um mesmo meio o que leva ao fenômeno de interferência, por exemplo, roteadores 802.11, telefones sem fio e microondas. Tendo este problema em vista, foi regulamentado que a potência de transmissão máxima destes dispositivos não seja superior a 1 watt. Além disso, técnicas de transmissão de dispersão do espectro de frequência tem sido utilizadas para que este problema seja minimizado (TANENBAUM; WETHERALL, 2010)).

A figura 5 ilustra diversos padrões de redes *wireless*, em função da velocidade de transmissão (*data rate*) e alcance (*range*). Dentre os padrões WLAN e WPAN, o 802.11 (WiFi) possui uma maior velocidade e alcance quando comparado com os padrões 802.15.1 (*Blueetooth*) e 802.15.4 (*Zigbee*).

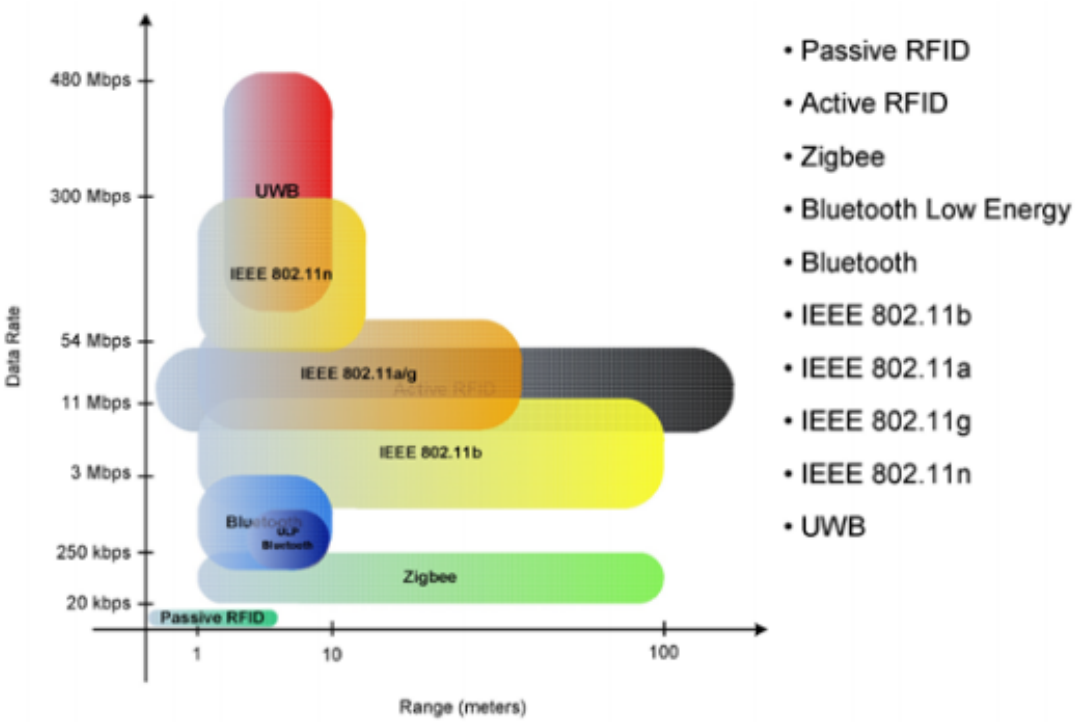

Figura 5: Velocidade e Alcance de padrões *Wireless* (DEAN, 2009)

2.2.3 SISTEMAS EMBARCADOS

Um sistema embarcado, de acordo com o autor Noergaard (2005), de uma forma geral, apresenta as seguintes características:

- É projetado, primariamente, para resolver uma função específica. Embora atualmente, sistemas embarcados possuem múltiplas funcionalidades, por exemplo, um *smartphone*.
- Possui mais limitações de *software* e *hardware* que um computador de uso pessoal devido as limitações financeiras de projeto.
- Requisitos que exigem uma alta qualidade e confiabilidade são necessários, se o sistema embarcado for projetado para lidar com uma situação crítica, por exemplo, o controle do sistemas de freios ABS de um carro.

Atualmente, o avanço de tecnologia proporciona que microcontroladores, com mais recursos de *hardware*, sejam desenvolvidos com um preço mais acessível. Consequentemente, novas funcionalidades podem ser agregadas ao sistema embarcado. De acordo com o autor, Sally (2009), existe uma ampla quantidade de sistemas embarcados utilizando sistema operacional, devido a facilidades que este proporciona para a construção de aplicações mais complexas. Um exemplo destas facilidades seriam, a disponibilidade de *device drivers*, pilha TCP/IP e *multitasking*.

2.3 CONSIDERAÇÕES FINAIS

Com o estudo realizado na fundamentação teórica foi possível verificar quais tecnologias se encaixam melhor ao projeto, as escolhas tecnológicas e os fundamentos serão apresentadas a seguir.

2.3.1 FUNDAMENTOS

Para o transporte de dados deste projeto foi definida a utilização da suite TCP/IP. De acordo Tanenbaum e Wetherall (2010), nesta suite existem dois protocolos de transporte amplamente utilizados TCP e UDP, sendo respectivamente estes protocolos, orientado a conexão e não orientado. Devido a esta característica o TCP consegue prover uma conexão confiável, através de mecanismos de controle de fluxo e congestionamento e retransmissões de pacotes. Para a transmissão dos pacotes de vídeo em tempo real, é amplamente utilizado

o protocolo UDP tendo em vista que este protocolo proporciona um menor *overhead* de transmissão, em relação ao TCP (TELSTRA, 2000).

Para a realização da transmissão de vídeo do ambiente de monitoramento à interface final do usuário, com o intuito de diminuir a quantidade de *bits* necessários para a transmissão, foi utilizado um algoritmo de codec, a escolha foi baseada em testes práticos onde foi analisado o consumo de banda, de processador e de memória de cada algoritmo, esse consumo foi analisado no processador escolhido para o desenvolvimento do projeto. Os algoritmos analisados foram H.264, MPEG-4 e JPEG (sem compressão) e será melhor detalhada no capítulo 4.

Visando a necessidade de realizar o controle de velocidade dos motores, para que o robô possa andar em linha reta, foi utilizado o algoritmo de controle PID. O controlador PID é definido pela equação 1, segundo o autor Ang et al. (2005), onde as funcionalidades atreladas a estas constantes serão descritas a seguir:

1. K_P: é a constante de ganho proporcional, a qual proporciona um controle geral do sistema, através do erro do sinal multiplicado pelo ganho. Todas as frequências envolvidas serão afetadas.

2. K_D: é a constante de ganho derivativo, a qual proporciona uma melhor reposta transitiva através de compensação derivativa de alta frequências.

3. K_i: é a constante de ganho integral, a qual proporciona uma redução no erro de regime estacionário através de compensação integral de baixas frequências.

$$G(s) = K_P + K_i\frac{1}{s} + K_D s \tag{1}$$

2.3.2 TECNOLOGIAS

Neste projeto, devido a necessidade de mobilidade para o robô é necessário utilizar uma tecnologia de transmissão de dados sem fio, a qual tenha uma velocidade de *upstream* suficiente para transportar vídeo em tempo real [1]. A banda necessária para a transmissão de um vídeo com resolução 640x480 e 30 FPS, utilizando a codificação H.264, é de 300 a 800 Kbps (MICROSOFT, 2013).

[1] Neste documento o termo tempo real significa uma taxa de transmissão de vídeo, na qual o olho humano perceba a sequência de imagens como um vídeo contínuo.

Tendo em vista estas necessidades, a tecnologia mais adequada ao projeto é 802.11. O padrão 802.11g possui uma taxa de transmissão de dados de 6 a 54 Mbps (IEEE, WORKING GROUP, 2007), podendo variar para menos de acordo com a quantidade de clientes 802.11g. Embora exista esta variação, a velocidade será suficiente para realizar o transporte do vídeo do ambiente.

Uma outra característica interessante que o 802.11 pode proporcionar a este projeto é a mobilidade através de uma topologia *mesh*, que consiste em diversos roteadores 802.11 com enlaces entre si (de forma parcial ou total) para possibilitar uma área maior de cobertura da rede 802.11, consequentemente, permitindo uma maior mobilidade para o cliente conectado a esta rede, que neste caso é o robô (CISCO, 2008).

Visando a facilidade de monitoramento do ambiente. a câmera, que está acoplado ao robô, tem a liberdade de dois ângulos de rotação, sendo assim, capaz de em uma determinada posição filmar diversos ângulos. Essa rotação é feita através de dois servo motores acoplados entre si. O interfaceamento com o sistema embarcado é via USB, pelo amplo número de câmeras que apresentam essa interface.

Com a utilização do padrão 802.11, faz-se necessário que o sistema embarcado tenha a implementação da pilha TCP/IP. Este também realiza o interfaceamento da câmera, por *device drivers* e, através de bibliotecas, realiza o processamento de imagens e codificação de vídeo. Uma escolha apropriada para o sistema embarcado, é um capaz de suportar um sistema operacional embarcado e tenha interfaces suficientes para realizar a integração com o sistema. Por ser um sistema operacional de distribuição gratuita e apresentar todas as características necessárias para o projeto, o mais adequado é o Linux, onde a linguagem de programação utilizada é o Python.

Por ser um sistema de monitoramento remoto, uma interface se faz necessária. Esta disponibiliza ao usuário um modo para monitorar o ambiente, enviar os comandos de direção ao robô e rotação da câmera. Para trazer uma maior liberdade ao usuário, em relação a qual sistema operacional aplicação irá executar, a interface remota é multiplataforma. Para que isso possa ocorrer a linguagem em que será programada foi o Java.

3 METODOLOGIA

Para o desenvolvimento do sistema foi necessário uma metodologia, esta foi divida em módulos onde cada um contempla uma parte do projeto, totalizando três módulos, onde cada um destes teve tarefas associadas para a realização do mesmo. Os módulos serão descritos a seguir:

1. **Sistema embarcado:** neste módulo foi necessário o estudo do microcontrolador escolhido, como programá-lo e compreender os modos de execução. Além disso, como interfacear os periféricos, codificar vídeo e implementar o algoritmo de controle para que o robô possa deslocar-se em linha reta e realizar curvas.
2. **Sistema de comunicação:** neste módulo foi definido como os procolos de comunicação, que transportam vídeo e dados, foram utilizados entre o cliente e o robô.
3. **Interface do cliente:** neste módulo foi desenvolvido a interface em que o usuário tem acesso ao robô através de autenticação. Nesta é possível monitorar o ambiente e enviar comandos ao robô.

O quadro 2 apresenta as tarefas que foram executadas para a elaboração do projeto, no total foram divididos em 10 tarefas, cada tarefa está relacionada a um dos três módulos, esta relação é demonstrada no quadro 3, onde cada “X” marcado representa que tarefa está relacionada com o módulo.

Quadro 2: Tarefas do Projeto

Código	Tarefa
A	Modelar o Projeto
B	Elaborar a Monografia de TCC
C	Adquirir os Componentes
D	Realizar Testes de Validação
E	Estabelecer a Comunicação TCP
F	Transportar o Vídeo UDP
G	Interfacear os Servo Motores
H	Interfacear os Motores DC e Implementar o Controle
I	Realizar o codec do Vídeo
J	Realizar a Autenticação do Usuário

Quadro 3: Associação entre Módulos e Tarefas

Módulos/Tarefas	A	B	C	D	E	F	G	H	I	J
Sistema Embarcado	X	X	X	X		X	X	X	X	
Sistema de Comunicação	X	X	X	X	X	X				
Interface do Cliente	X	X		X					X	X

Em algumas tarefas ocorre a dependência de que a outra esteja terminada para que possa ser executada, porém algumas tarefas podem ser realizadas paralelamente uma da outra, a figura 6 demonstra como o projeto foi realizado.

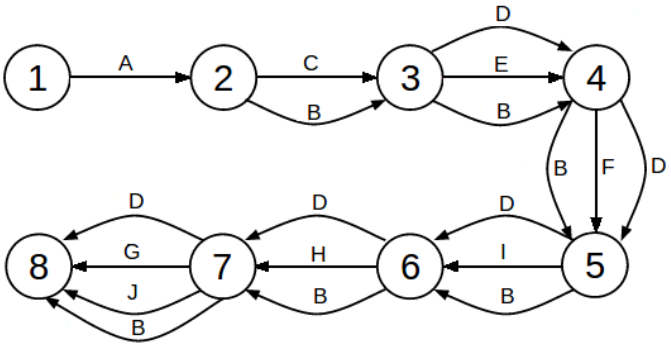

Figura 6: Etapas do Projeto (Autoria Própria)

3.1 RECURSOS

Nesta seção são apresentados os recursos de *hardware* e *software*, que foram utilizados nesse projeto.

3.2 HARDWARE

Os recursos de *hardware* necessários no escopo do projeto englobam, estrutura física, o sistema embarcado, o sistema de comunicação e um computador onde encontra-se a aplicação da interface de monitoramento e comandos. Todos os itens foram adquiridos por meio próprios. As subseções a seguir irão detalhar os recursos que compõem o projeto.

3.2.1 ESTRUTURA FÍSICA

Para a estrutura física é utilizado um chassis ROVER 5, essa escolha deve se ao fato que ao comprar um chassis pronto, não haveria gasto de tempo da equipe para a construção. Outro fator importante é o meio de locomoção dele ser por meio de esteira, esse modo faz com que ele possa andar em terrenos irregulares e com tipos diferentes de pisos. Os motores do robô são ligados através de seis pilhas AA. O robô apresenta as seguintes configurações:(ROBOTICS; ELECTRONICS, 2013)

- Dois motores de corrente contínua de pico de 2,5A e tensão 7,2V.
- Dois *encoders* com resolução de 1000 pulsos por 3 revolução.

Ainda na parte da estrutura física foi utilizado dois servo motores, para fazer a rotação da câmera, esta tem resolução de 16MP interpolados. Além disso, esta câmera possui LEDs o que possibilita que ambientes não totalmente iluminados possam ser monitorados, o modo de interfaceamento é através de USB.

3.2.2 SISTEMA DE COMUNICAÇÃO

O sistema de comunicação é composto por uma antena conectada via USB, a antena opera na frequência 2,4GHz, e suporta o protocolo 802.11g. No ambiente onde o robô estiver monitorando faz-se necessário um roteador com suporte ao protocolo 802.11g.

3.2.3 SISTEMA EMBARCADO

O sistema embarcado é composto por uma placa BeagleBone, com ela é possível utilizar um sistema operacional embarcado Linux. Segue as especificações (COLEY, 2012):

- Um processador ARM Cortex-A8 AM3359 - 720Mhz.
- Memória 256MB DDR2 RAM.
- Uma interface *ethernet*.
- Uma interface USB, por ser necessário duas interfaces, para câmera e antena, será utilizado um USB-*port hub*.

Para ligar a placa mais seus periféricos é utilizada uma bateria LI-PO de 4400mAh e 11.1V. Onde é utilizado um regulador de tensão para que esta possa ir a 5V.

3.2.4 INTERFACE DE MONITORAMENTO E COMANDOS

Para executar a aplicação é necessário um computador, desktop ou portátil, que tenha disponibilidade de trocar informações com o roteador.

3.3 SOFTWARE

Os recursos de *software* são:

- Biblioteca Gstreamer : utilizado para realizar codificação e decodificação de vídeo. Esta apresenta uma licença LGPL.
- Java: linguagem utilizada para programar a interface remota.
- Astah community: utilizado para realização de diagramas UML.
- Wireshark: utilizado para analisar os pacotes de redes.
- Eclipse: utilizado para realizar a programação da interface remota
- Python: linguagem utilizada para realizar a programação do *software* do sistema embarcado.

4 DESENVOLVIMENTO

Para o desenvolvimento do projeto foi utilizado como base a figura 7, que ilustra a arquitetura do sistema através de um diagrama de blocos. Nesta é possível verificar que o sistema é composto por três blocos principais, a Interface de Monitoramento e Comandos (IMC), o Sistema Embarcado (SE) e a Infraestrutura de Rede. Como é utilizado o sistema operacional Linux no SE, foi possível realizar testes simulados inicialmente em um computador com Linux e depois foi passado para a BeagleBone, desta forma diminuindo os risco de a placa ser danificada.

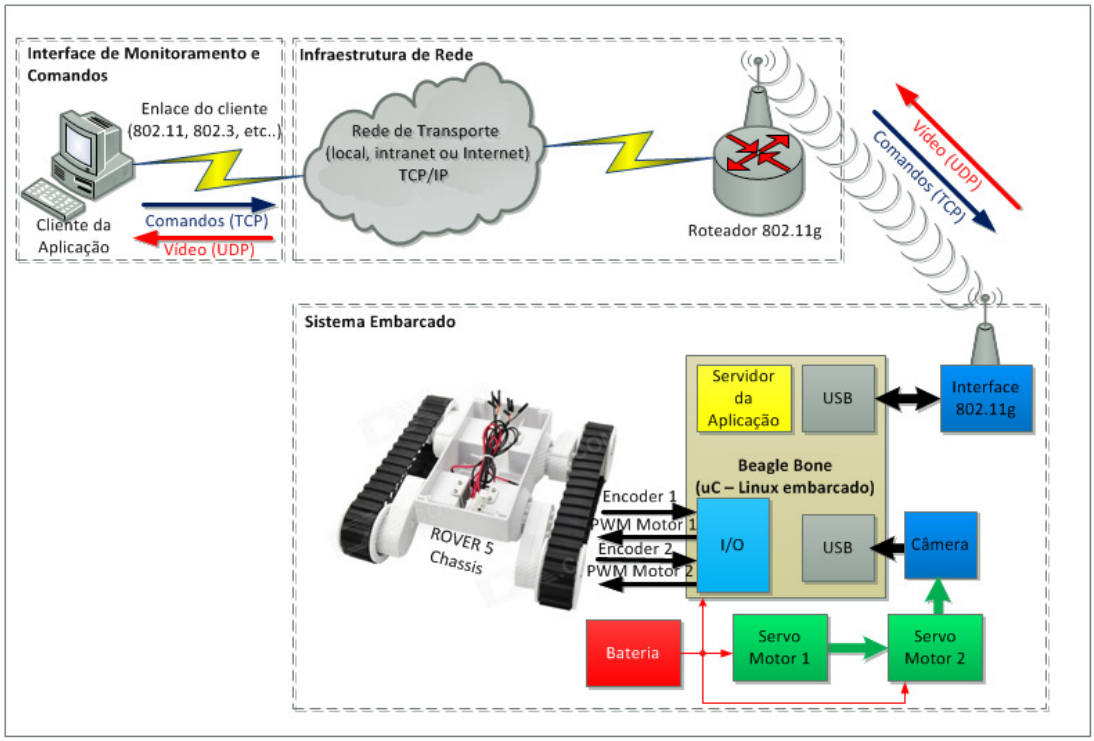

Figura 7: Diagrama de Blocos do Sistema (Autoria Própria)

4.1 SISTEMA DE COMUNICAÇÃO

O sistema de comunicação IP foi divido em duas partes, transmissão de dados e de vídeo. A infraestrutura de rede, a qual possibilita conectividade entre a IMC e o SE, pode conter um ou diversos domínios de rede. Em ambos os casos, a transmissão realizada entre o robô e o roteador é efetuada através de antenas *wireless* 802.11g. O roteador que fornece conectividade para o SE deve apresentar a configuração conforme ilustrado nas figuras 8 e 9. A transmissão da IMC à infraestutura de rede, pode utilizar um enlace 802.3 ou 802.11. Além disso, o SE sempre terá um IP privado atribuído de forma estática, devido a um problema de versão do *driver* utilizado da antena em relação a utilização de DHCP, conforme está ilustrado na figura 8 e 9. A figura 8 representa o cenário onde existe apenas um domínio. Neste caso, conforme ilustrado, a IMC realiza uma *query* DNS para o roteador, com o objetivo de obter o IP atribuído ao SE e consequentemente estabelecer a sessão TCP.

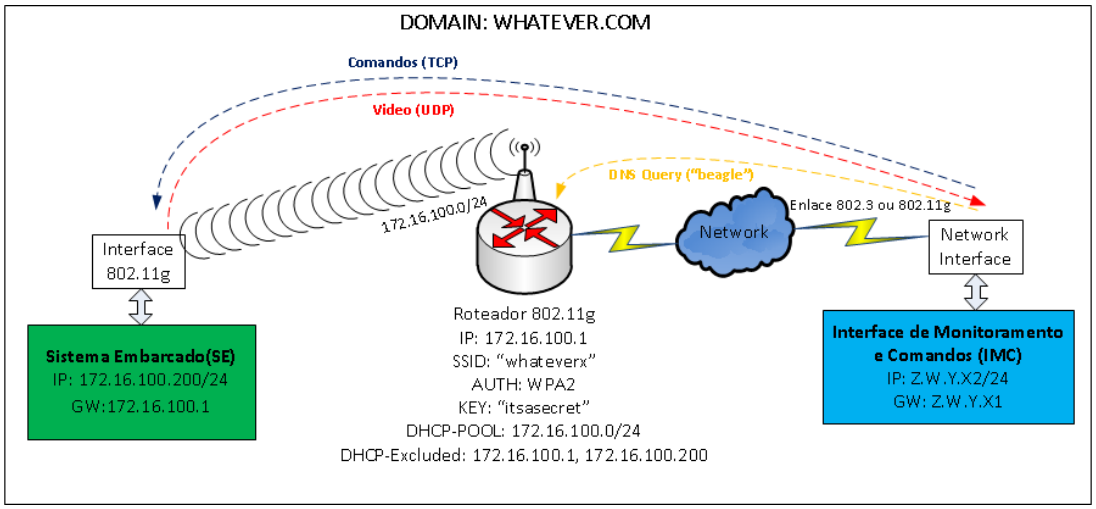

Figura 8: Infraestrutura de rede com apenas um domínio (Autoria Própria)

No caso da infraestrutura de rede conter diversos domínios, faz-se necessário que exista um servidor de DNS para que a IMC possa obter o IP atribuído ao SE. Um exemplo de como utilizar um servidor de DNS, de forma gratuita, está ilustrado na figura 9. Além disso, devido aos diversos domínios, é necessário que os roteadores de borda, onde se encontram a IMC e SE, façam a tradução dos endereços de IP público para privado através de NAT. Na figura 9, é utilizado o serviço do site NOIP, este fornece o serviço de DNS dinâmico, com ele é possível criar um registro em um determinado domínio para o robô. Posteriormente, após a criação do registro, para que a resolução do registro para o IP público do roteador de borda mantenha-se coerente, o SE deve utilizar o *software* do NOIP para que exista uma atualização periódica através de DNS *update*. A *query* DNS, que no exemplo anterior era enviada para o roteador, neste caso, posteriormente, deve ser redirecionada para o servidor do NOIP. Por fim, a sessão

TCP é estabelecida através dos IPs públicos dos roteadores de borda.

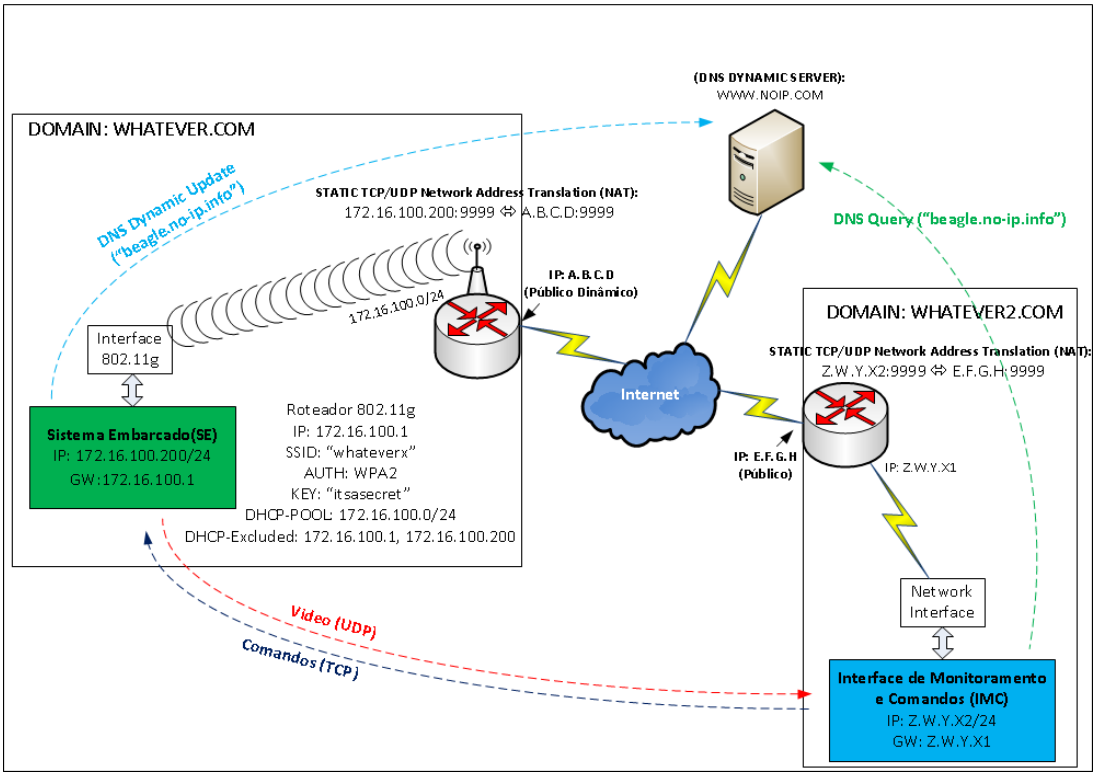

Figura 9: Infraestrutura de rede com diversos domínios (Autoria Própria)

A transmissão de dados foi realizada através do protocolo de transporte TCP, que garante a entrega dos pacotes, utilizando a porta 9999. Para a comunicação ocorrer foi implementado um servidor TCP no *software* do SE e um cliente TCP na IMC, ambos utilizam bibliotecas nativas. Para codificar uma semântica entre a IMC e o SE, foram estabelecidos quatros codificações de pacotes, o de comando, autenticação, alteração de senha e chave de sessão. A semântica de codificação está apresentada no quadro 4. Cada codificação de pacote, apresenta três caracteres especiais para identificar seu tipo. O caracter "!" é utilizado como delimitador entre o conteúdo do pacote e o seu fim. Devido a utilização do delimitador, o conteúdo dos pacotes pode ser variável e deve conter somente os caracteres [(a-z) (A-Z) (0-9)].

Quadro 4: Associação entre Módulos e Tarefas

Tipo de Pacote	Exemplo
Comando	+++!Comando!!!!!
Autenticação	###!Senha!!!!!
Alteração de Senha	***!SenhaNova!!!!!
Chave de Sessão	&&&!ChaveSessão!!!!!

O quadro 5 apresenta os comandos e seus respectivos significados que são enviados da IMC para o SE. O tamanho dos comandos foi definido como sendo de apenas 1 byte,

pois existem poucos comandos, sendo assim, facilitando o *parsing*. O comando “IDLE”, apresentado no quadro, é utilizado como uma forma de verificação da conexão, um *keep alive*. Caso passe mais de 2 segundos sem o usuário enviar algum comando, o “IDLE” é enviado, caso não houver uma resposta o usuário é informado que houve uma queda na conexão TCP e este deve tentar se reconectar. Todos os pacotes que são enviados através do TCP, são cifrados com um algoritmo de Cesar, onde cada caracter é substituido por um outro de acordo com o valor da chave. Sendo que este valor é definido no início da conexão, e persiste até a próxima troca de chave, através do pacote Chave de Sessão, onde o tamanho da chave varia entre 1 e 21. Mesmo não sendo este um algoritmo robusto de criptografia, o mesmo atende a necessidade de confidencialidade, tendo em vista que este projeto não é um sistema crítico.

Quadro 5: Codificação dos Comandos

Dados	Comando
0	Transmitir Vídeo
1	Para Vídeo
2	Andar em frente
3	Virar a Esquerda
4	Virar a Direita
5	Mover Câmera Acima
6	Mover Câmera Abaixo
7	Mover Câmera Direita
8	Mover Câmera Esquerda
9	IDLE
:	Transmissão Baixa Qualidade
;	Inicializar
¡	Mover 1 segundo em frente
=	Mover 2 segundos em frente
¿	Mover 3 segundos em frente
?	Mover 4 segundos em frente
@	Virar por 1 segundo
A	Virar por 2 segundos
B	Virar por 3 segundos

A transmissão do vídeo é feita através do protocolo UDP, porém para a transmissão foi utilizado a biblioteca GSTreamer, onde esta é responsável por, além de realizar o codec do vídeo, enviar o vídeo para a IMC. Nela é definido o IP de destino e a porta de conexão, e também a porta utilizada, que neste acaso foi a mesma do TCP 9999, isto foi possível pois são dois protocolos diferentes. A diferença em relação ao TCP é que o servidor agora encontra-se na IMC, e não mais no SE, a figura 10 apresenta esta comunicação entre as duas pontas.

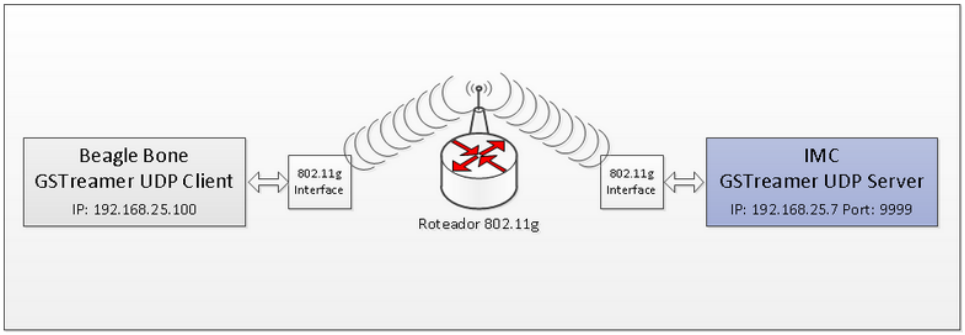

Figura 10: Transmissão UDP, entre SE e IMC (Autoria Própria)

4.2 HARDWARE

Pela necessidade de interfaceamento dos componentes de *hardware* foi necessário elaborar um circuito, a figura 11 demonstra um diagrama em blocos do *hardware*. Este contempla todos os interfaceamentos necessários para a conexão dos motores, servos e *encoders*.

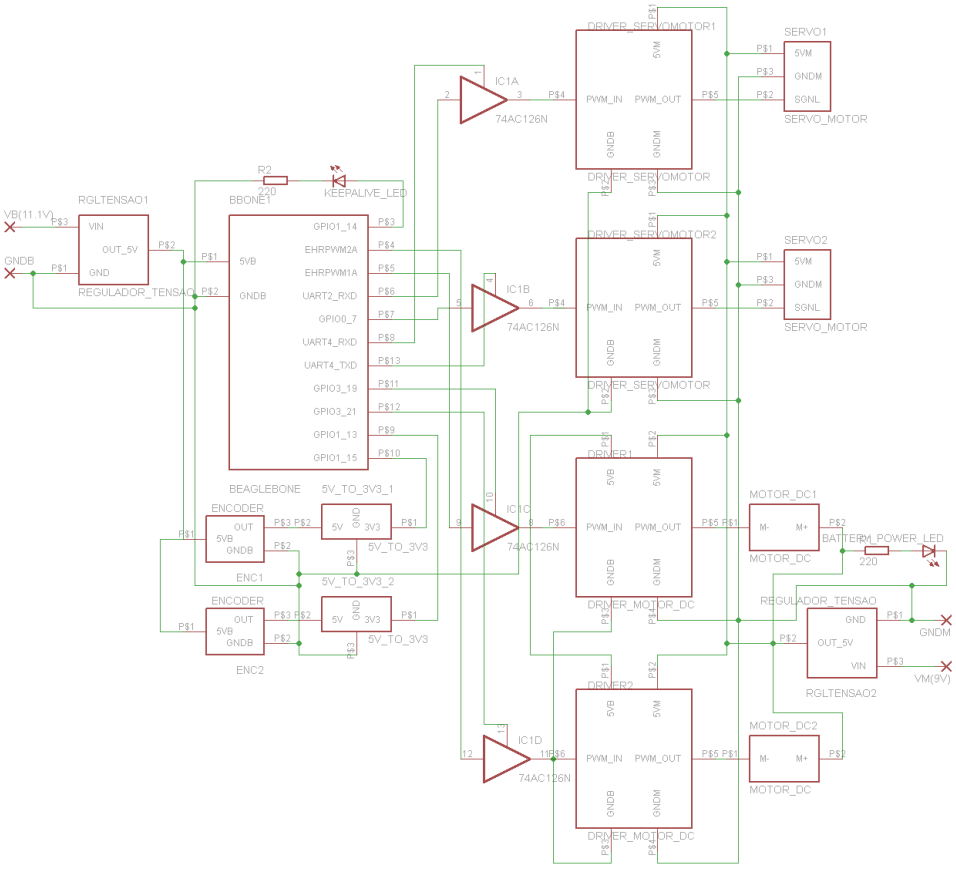

Figura 11: Diagrama de Blocos do Hardware (Autoria Própria)

O diagrama elétrico que implementa o diagrama em blocos está ilustrado na figura 12.

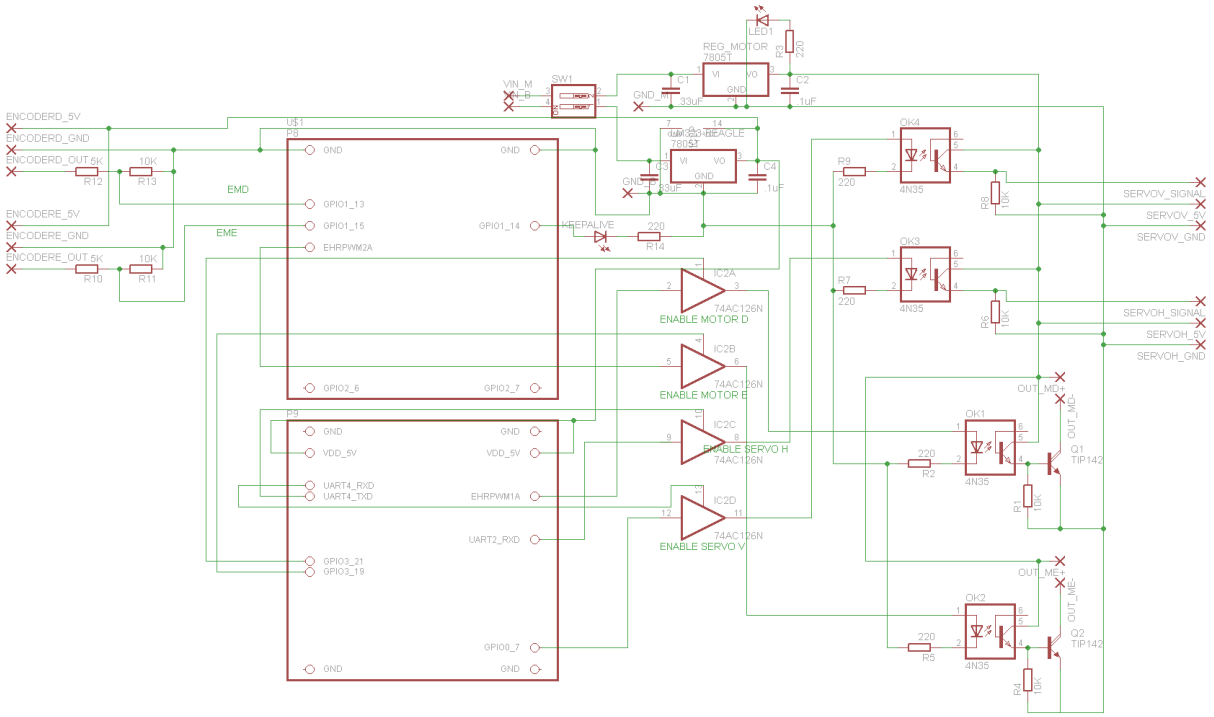

Figura 12: Diagrama Elétrico Hardware (Autoria Própria)

Para realizar o interfaceamento do *encoders*, foi necessário utilizar um circuito de divisão de tensão de 5V para 3.3V, conforme mostra a figura 13, pois o conforme do *datasheet* da placa BeagleBone (COLEY, 2012), os pinos de *input* tem uma tensão máxima de entrada de 3.3V.

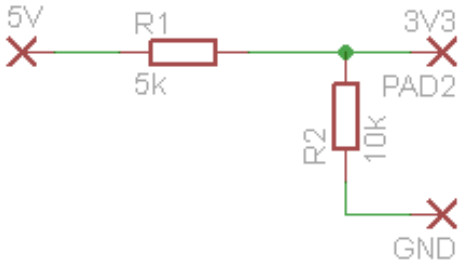

Figura 13: Divisor de Tensão 5V para 3V3 (Autoria Própria)

No projeto é utilizado duas formas de bateria, uma que liga somente *encoders*, placa e seus periféricos, uma bateria LI-PO, com capacidade de 4400mAh e 11.1V, sendo que o máximo utilizado pelo circuito é 2A, podemos analisar que bateria durará aproximadamente 2 horas, e outra somente para ligar os motores e servos, essa formada por 6 pilhas AA em série de 1.5V. Como ambas baterias necessitam que a tensão seja de 5V, foi utilizada dois reguladores de tensão, conforme a figura 14. Para interfacear os motores e servos foi necessário

criar *drivers* de corrente, a figura 15 demonstra o interfaceamento dos motores onde é utilizado um fotoacoplador, desta forma eliminando os ruídos que seriam causados pelos motores, já que este é uma carga indutiva.

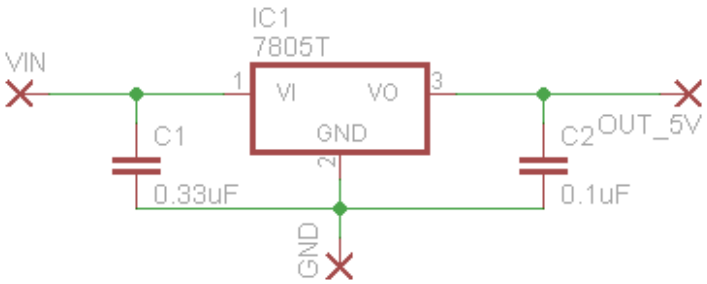

Figura 14: Regulador de Tensão (Autoria Própria)

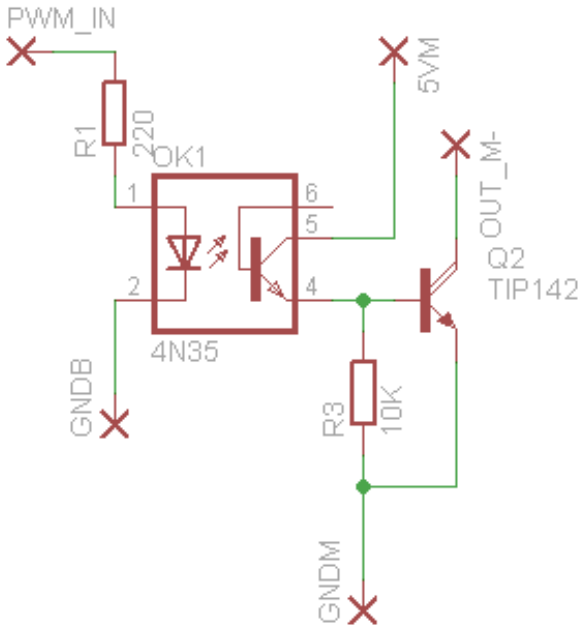

Figura 15: Interfaceamentos Motores (Autoria Própria)

Os servos também são alimentados com uma tensão de 5V e possuem um fotoacoplador, pois sua carga indutiva causa ruído no circuito, a figura 16 demonstra o interfaceamento. Tanto os servos e motores apresentam *buffer tri-state*, devido ao fato que quando o robô era ligado existia um problema crônico de habilitação dos mesmos, com a colocação dos *buffers* isto não ocorre mais. Todos os *buffers*, tem como entrada um sinal de PWM, para que seja possível controlar os motores DC e os servo motores.

Existem dois LEDs no projeto, um que ficará ligado quando a bateria do motores estiver conectada ao circuito, e outro somente para acender e apagar a cada *keep alive* ou comando recebido da IMC, com isto é possível saber se o robô está ou não conectado com

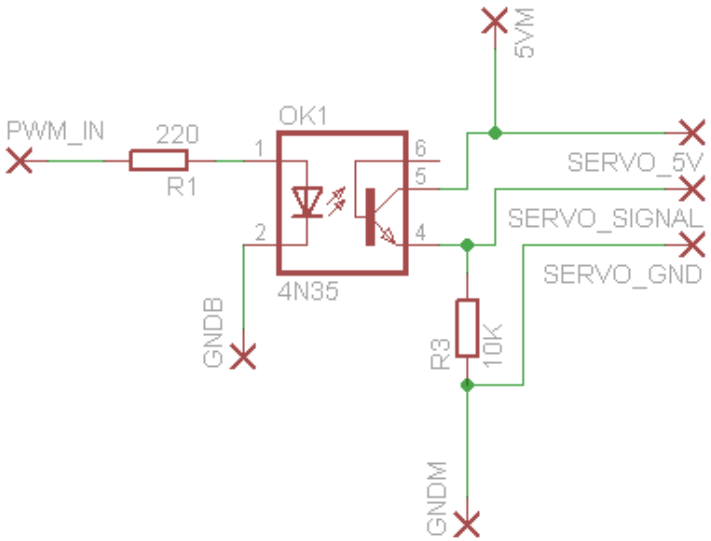

Figura 16: Interfaceamento Servos (Autoria Própria)

a IMC. Existem ainda dois botões para ligar o robô, um que liga somente os *encoders* e a BeagleBone e seus periféricos, o outro para ligar os motores e servos.

Após o projeto de *hardware* ser realizado, o próximo passo foi realizar a PCB do diagrama, demonstrado na figura 17. Esta foi realizada em *dual-layer* pois, como visto na figura, o roteamento das trilhas não foi possível em apenas uma camada. A camada top é a em vermelho, e a down em azul. A figura 18 apresenta como ficou a PCB que foi elaborada e construída manualmente.

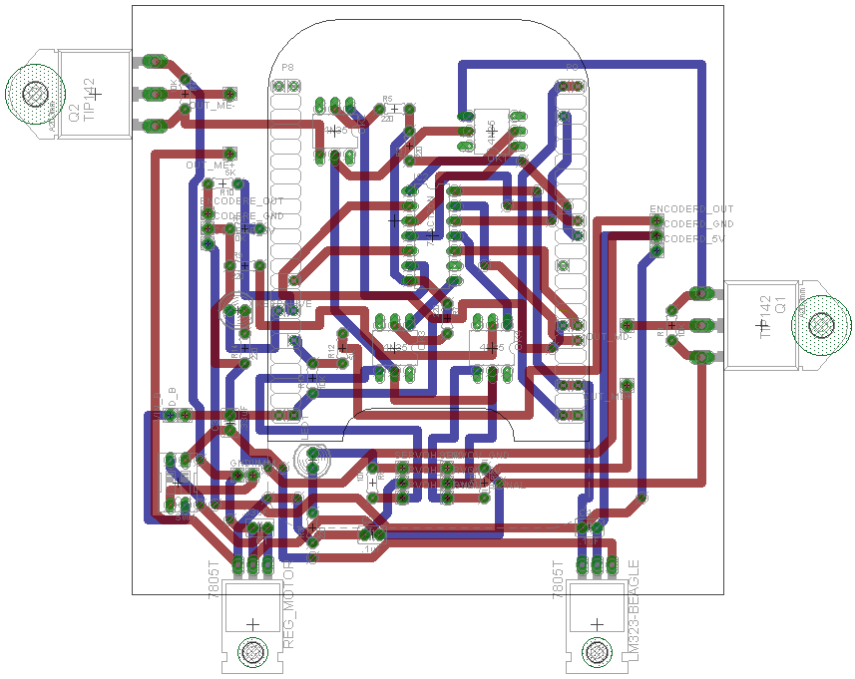

Figura 17: Diagrama PCB (Autoria Própria)

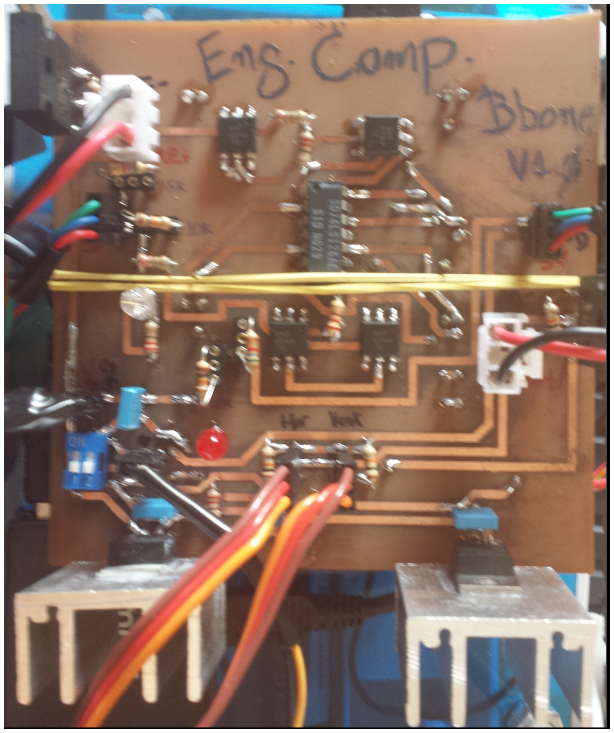

Figura 18: PCB modelo final (Autoria Própria)

4.3 SOFTWARE SISTEMA EMBARCADO

Inicialmente para iniciar o projeto do *software* do sistema embarcado, foi alterado o sistema operacional da placa BeagleBone de angstrom para o Ubuntu, pois os *drivers* dos periféricos disponibilizados mais recentemente era pra o Ubuntu, após instalação do Ubuntu, foi instalado os *drivers* da câmera, da antena e do USB-*port hub*. Após instalados, foi necessário verificar os modos de acesso aos pinos de *input* e *output*, o acesso a estes é realizado através de escrita/leitura em *file descriptors*, para habilitar o pino é necessário utilizar um comando "export" do Linux. O resultado do comando export cria um diretório para utilizar o pino, onde encontra-se os arquivos necessários para o funcionamento do mesmo.

Os pinos de entrada e saída (I/O) apresentam dois arquivos, o *direction*, onde é definido se o pino é de entrada ou de saída escrevendo *in* ou *out* respectivamente, e caso for pino de saída é necessário escrever seu valor no arquivo *value*, sendo 1 ou 0. De forma análoga, é realizado leitura se o pino for de entrada. Os pinos nos modos PWM apresentam outra estrutura de arquivos onde, o arquivo *period* define o período da onda, *duty* que define o ciclo ativo da onda e o *run* onde é definido se o PWM está ligado ou desligado, escrevendo 1 ou 0. Cada vez que a BeagleBone é desligada os diretórios dos pinos são apagados, com isso foi necessário criar um script de inicialização, que é executado após o *boot* da placa, para automatizar a configuração dos pinos utilizados pelo projeto.

O *software* do sistema embarcado foi escrito em Python, e contém os seguintes requisitos funcionais e não funcionais.

Requisitos Funcionais

RF1 - O *software* deverá executar comandos oriundos da IMC.

RF1. 1 - O *software* deverá receber o comando através da interface de rede.

RF2 - O *software* deverá decifrar todo o pacote de dados recebido.

RF3 - O *software* deverá cifrar todo o pacote de dados enviado.

RF4 - O *software* deverá enviar vídeo do ambiente para a IMC.

RF4. 1 - O *software* deverá realizar a compressão do vídeo.

RF5 - O *software* deverá realizar o controle de velocidade dos motores.

RF5. 1 - O *software* deverá ler os *encoders*.

RF6 - O *software* deverá autenticar o usuário através de senha, oriunda da IMC.

RF7 - O *software* deverá alterar a senha do usuário.

RF8 - O *software* deverá permitir apenas uma conexão.

Requisitos Não Funcionais

RNF1 - O vídeo deverá ser enviado em tempo real.

RNF2 - O *software* deverá enviar o vídeo utilizando o protocolo UDP.

RNF3 - O *software* deverá utilizar sistema operacional Linux.

RNF4 - O controle de velocidade será realizado através do algoritmo PID.

O *software* apresenta três threads, Servidor TCP, Controle PID e Timers Motor, e um Sub Processo que é responsável por enviar as imagens obtidas da câmera. A biblioteca utilizada para realizar a captura da imagem, realizar o codec e transmissão de vídeo foi GSTreamer. Na qual é possível configurar diversos parâmetros de configuração de cada algoritmo de codec, inicialmente foi pensando em enviar imagens no tamanho de 640x480, porém foi verificado que o microcontrolador não tem capacidade de processamento para tal configuração, tendo em

vista que a biblioteca utiliza a codificação em *software*. Devido a esta limitação, foi necessário reduzir a resolução do vídeo para 320x240.

Com o valor de tamanho da imagem definido, foi iniciado o teste do melhor algoritmo de codec, o primeiro teste foi o H.264, pois este conforme visto no referencial teórico, apresentava uma maior eficácia de compressão e com isso utilizaria menor banda da infraestrutura, mas ao realizar teste foi possível perceber que a imagem que o usuário recebia não era boa suficiente, pois qualquer alteração brusca na imagem, demorava para que a mesma pudesse ser visualizada corretamente. Isto ocorre devido ao algoritmo que realiza uma previsão da imagem sucessora, através das imagens anteriores, por isso uma alteração de uma imagem brusca para outra demora para que a mesma seja visualizada corretamente. Para que o usuário tenha uma melhor visão do ambiente foram realizados alguns testes com os algoritmos MPEG-4 e JPEG sem compressão, para ver se os recursos utilizados a mais eram muitos maiores e com isso não sendo possível a execução.

As análises foram feitas utilizando o *software* wireshark com a topologia apresentada na figura 10, levou-se em conta principalmente o consumo de banda, mas também a utilização do processador e memória. A figura 19, apresenta um gráfico comparativo entre o JPEG e H.264, neste gráfico foram utilizados cinco tipos diferentes de configuração dos parâmetros do H.264 para verificar qual configuração apresentava melhor resultado ao usuário, a que apresentou melhores resultados para o usuário foi a configuração do H.264#4, em relação ao JPEG foi notado que sem a compressão o *payload* é grande isso faz com que seja necessário a fragmentação de pacotes, e através disso o *header* do UDP não estará mais presente, portanto, dificultando políticas de QoS que normalmente classificam as aplicações através do protocolo de transporte e portas.

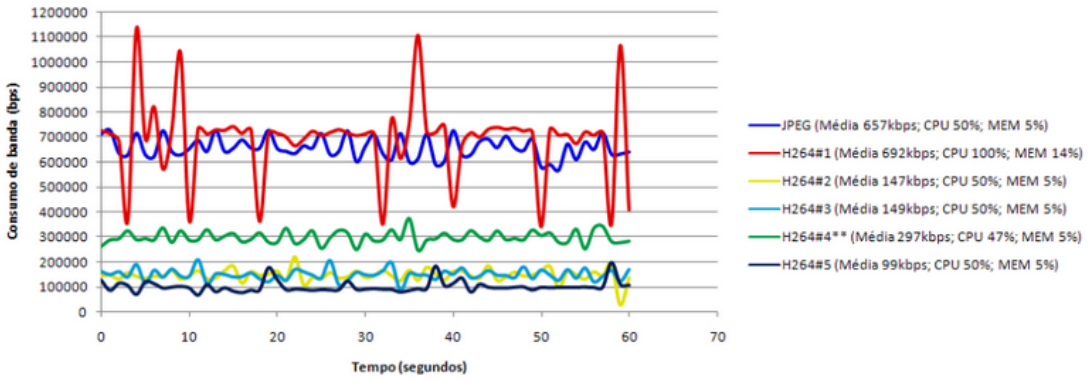

Figura 19: Comparação entre algoritmos H.264 e JPEG (Autoria Própria)

A figura 20 apresenta o gráfico de comparação entre o JPEG e duas configurações do algoritmo de codec MPEG-4, analisando as duas configurações é possível verificar que a única

diferença está no consumo de banda, 31Kbps a mais para a configuração MPEG-4#2, porém se analisarmos o aspecto visual foi esta que apresentou melhores resultados. A figura 21 demonstra o gráfico comparativo das melhores configurações dos algoritmos de codec e o JPEG, é possível analisar que o consumo de banda do H.264 é maior do que o MPEG-4, porém seu consumo de processamento é menor. Entretanto, é importante ressaltar que talvez um "tunning" fino dos parâmetros do H.264 pudesse reverter a situação. A escolha ficou para utilizar o MPEG-4 como algoritmo de codec, pois seu resultado visual ficou melhor do que o H.264, caso o usuário tenha um disponibilidade de rede maior e queria ver as imagens melhores, é possível que usuário escolha JPEG, porém utilizará mais recursos da rede.

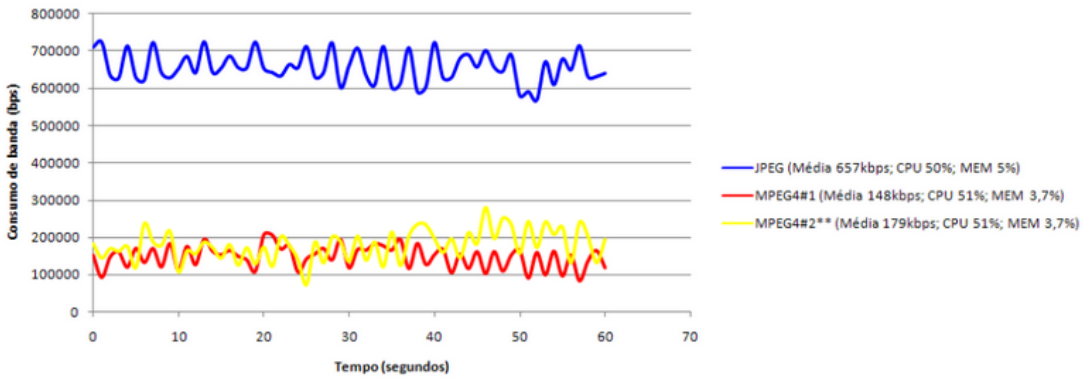

Figura 20: Comparação entre algoritmos MPEG e JPEG (Autoria Própria)

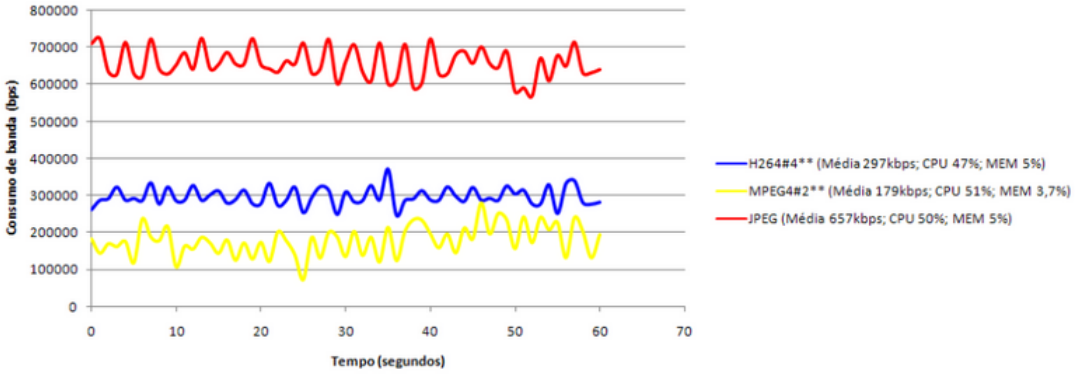

Figura 21: Comparação entre algoritmos H.264, MPEG e JPEG (Autoria Própria)

A thread servidor TCP é a responsável por analisar a chegada dos pacotes na porta 9999, caso ocorra, esta decifra os pacotes, através do algoritmo de Cesar e os decodifica. Antes que qualquer comando possa ser executado, é necessário que o usuário esteja autenticado. A autenticação ocorre quando a senha enviada, através da IMC, é decifrada e verificada se está de acordo com a senha no servidor TCP. Após estar autenticado os comandos passam a ser executados. Quando é recebido um pacote de alterar senha, inicialmente é enviado novamente

um pacote informando a senha atual, e depois a nova senha, caso a senha atual não esteja correta, o procedimento de alterar senha é abortado.

Cada vez que um pacote de comandos é recebido, é verificado que tipo de pacote que está sendo recebido e executa o comando solicitado. Quando pacote "IDLE" é recebido, é acendido e apagado o LED de *keep alive*. Caso o comando for de iniciar a transmissão do vídeo, cria-se um sub processo, ou o mata, caso seja para parar a transmissão. Com relação aos servo motores estes apresentam graus de liberdades que irão ser movimentados, o servo que executa os movimentos horizontais tem a liberdade de se movimentar de 0 a 180 graus, o servo, que executa movimentos verticais, apresenta 110^{o} de liberdade, de 40^{o} a 150^{o}.

Quando um comando de mover o robô chegar, é iniciada a thread de Timers Motores que é a responsável por analisar quantos segundos os motores ficarão acionados, de acordo com que o usuário escolheu na IMC. Se o comando for de andar em frente é necessário realizar o controle, e por isso é chamado a thread Controle PID. Esta é responsável para realizar os cálculos do controlador PID, onde esta lê os *encoders* e verifica as diferenças de valores entre os dois motores, caso o valor seja diferente é calculado um novo valor de PWM para os motores. O cálculo do controle é pelo algoritmo PID, conforme mostrado no capítulo 2, onde os valores das constantes foram estabelecidas conforme teste prático, este valores são Kp = 2, Kd = 2 e Ki = 0.005. O controlador só é aplicado caso o usuário queria andar em linha reta, nos casos em curva, se for para direita o motor direito é desligado, no caso da esquerda, o motor esquerdo é desligado.

A figura 22 apresenta a máquina de estados do *software* embarcado, esta demonstra as execuções em paralelo que podem ocorrer, é possível verificar que o único modo de a máquina de estado desligar é quando ocorrer uma falta de energia, e que quando ocorrer uma queda na conexão TCP, é voltado ao estado de Esperando Conexão. A máquina de estados do servidor TCP, está melhor apresentado na figura 23, onde é possível analisar que nenhum comando será executado se antes não tiver ocorrido a autenticação. O envio do vídeo é mostrado na figura 24. A figura 25, é possível ver que toda a movimentação do robô é executada em paralela, durante os segundos definidos pelo usuário, sendo assim o usuário pode enviar outros comandos, não sendo de movimentação, que serão executados.

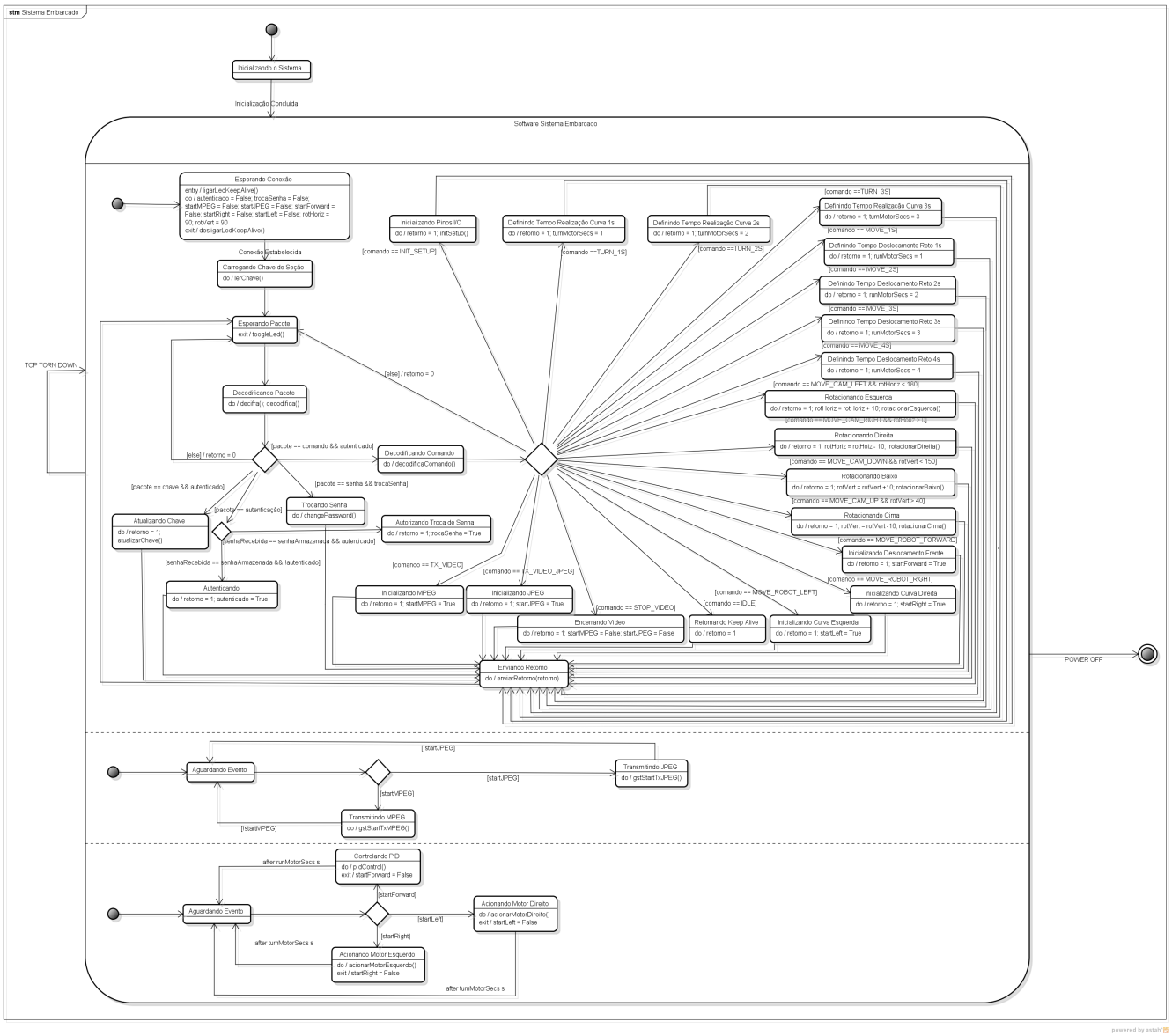

Figura 22: Máquina de Estado do Sistema Embarcado (Autoria Própria)

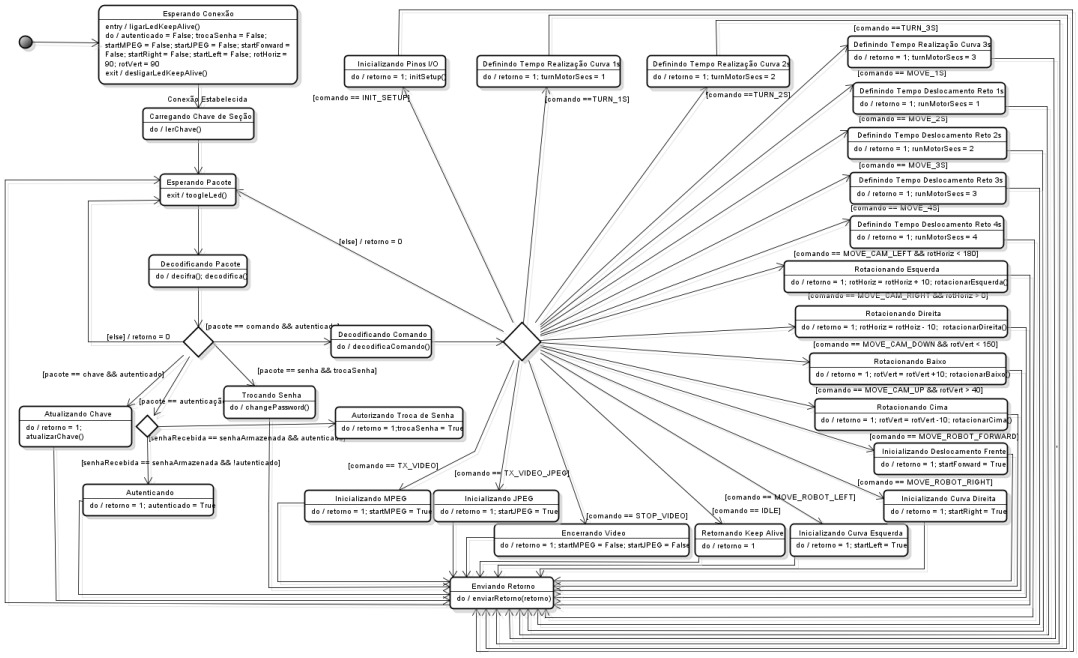

Figura 23: Máquina de Estado Servidor TCP (Autoria Própria)

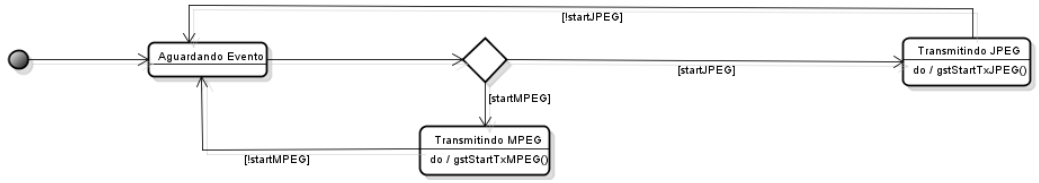

Figura 24: Máquina de Estado Transmissão de Vídeo (Autoria Própria)

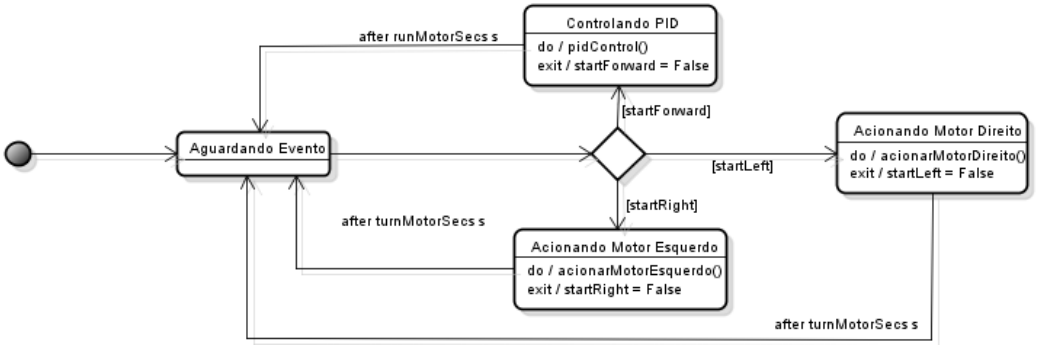

Figura 25: Máquina de Estado Movimentação Robô (Autoria Própria)

4.4 INTERFACE DE MONITORAMENTO E COMANDO

Inicialmente foi levantando os requisitos funcionais e não funcionais do *software*, e foi através dele que foi elaborado a interface, os requisitos são: Requisitos Funcionais

RF1 - O *software* deverá enviar comandos ao sistema embarcado.

RF1. 1 - O *software* deverá enviar o comando através da interface de rede.

RF1. 1. 1 - O *software* deverá garantir a entrega do comando.

RF2 - O *software* deverá cifrar todos os pacotes de dados enviados.

RF3 - O *software* deverá decifrar todos os pacotes de dados recebidos.

RF4 - O *software* deverá exibir o vídeo do ambiente monitorado.

RF4. 1 - O *software* deverá receber vídeo do sistema embarcado através de rede.

RF4. 2 - O *software* deverá realizar a descompressão do vídeo.

RF4. 3 - O *software* deverá exibir o vídeo em uma janela.

RF5 - O *software* deverá exibir uma mensagem de erro em caso de indisponibilidade de rede.

RF6 - O *software* deverá bloquear os movimentos do robô, enquanto este está sendo executado.

RF7 - O *software* deverá autenticar o usuário.

RF8 - O *software* deverá permitir a troca de senha.

RF9 - O *software* deverá mostrar os *log* do sistema.

Requisitos Não Funcionais

RNF1 - O *software* deverá exibir o vídeo em tempo real.

RNF2 - O *software* deverá enviar comandos utilizando o protocolo TCP.

RNF3 - O *software* deverá ser programado em Java.

RNF4 - A autenticação deverá ser realizada através de senha.

Com requisitos levantados a interface começou a ser programada, a linguagem escolhida foi Java. A figura 26 apresenta o diagrama de classes da IMC, nele é possível ver as três classes que possuem a IMC. A classe IMCGUI cria três threads a tGst, tMillis e tKeepAlive. A thread tGst é responsável por chamar o método run da classe GStreamer, onde esta irá receber e exibir as imagens enviadas do robô. Outra thread que pode ser visto é a do tKeepAlive, esta é responsável por ficar enviando o parâmetro "IDLE" para o sistema embarcado, assim desta forma consegue verificar se não ocorreu perda de conexão com o robô. A thread tMillis é apenas responsável por desabilitar os botões de movimento do robô pelo tempo definido pelo usuário, assim dessa forma não é possível que o usuário fique enviando comandos de movimentação do robô sem antes ter terminado o anterior. O usuário antes de se autenticar escolhe o tempo que o robô ficará andando em linha reta ou em curva, depois de se autenticar o usuário poderá utilizar-se da interface gráfica para comandar o robô, tanto na parte de iniciar ou parar a transmissão de vídeo, quanto os comandos de deslocamento e o posicionamento do servo.

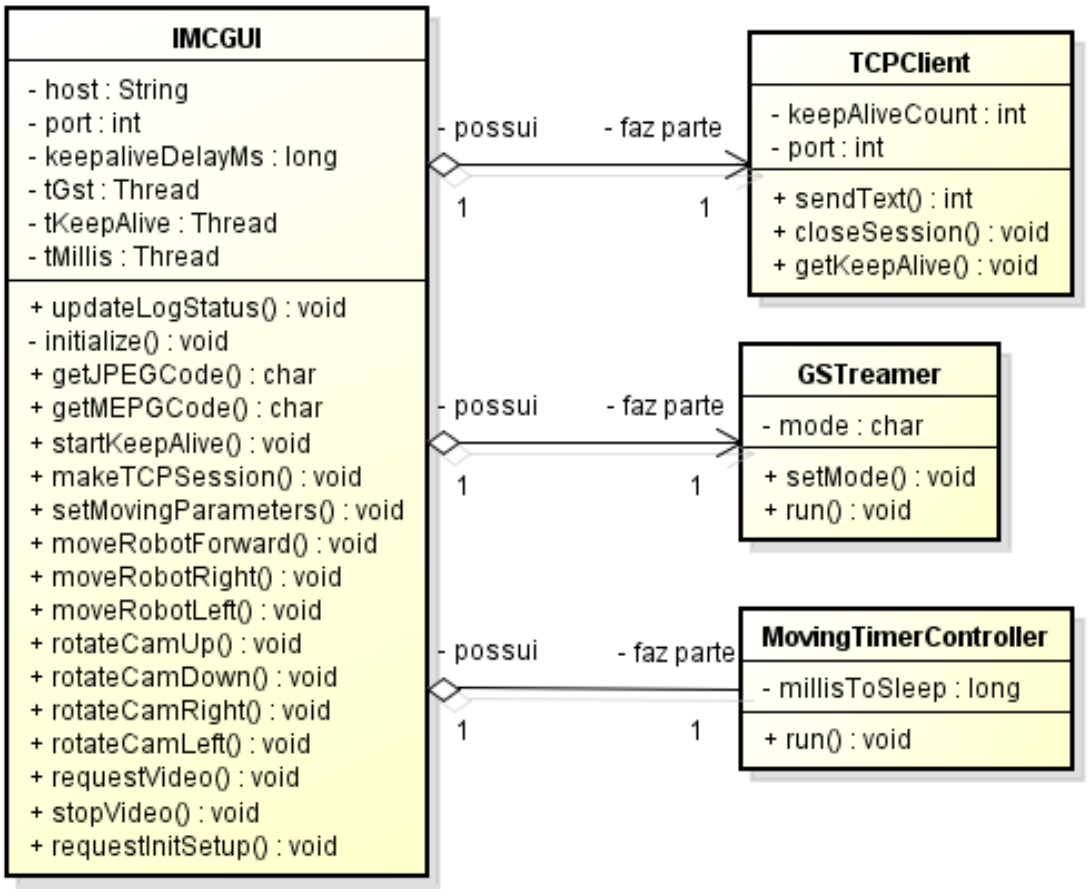

Figura 26: Comparação entre algoritmos H.264, MPEG e JPEG (Autoria Própria)

Todos os pacotes, para serem enviados, são cifrados com o algoritmo de Cesar e passados para o método sendText, da classe TCPClient, onde foi criado um cliente Servidor TCP, para que possa ser enviados os dados, este método retorna um inteiro, sendo este 1

ou 0, sendo 1 informando que o pacote enviado, foi recebido e executado com sucesso, e 0 quando ocorreu algum problema, isso pode ser quando o usuário informou a senha incorreta, ou tentou enviar um comando sem estar autenticado. Na classe GSTreamer foram utilizadas duas bibliotecas, GSTreamer, a mesma utilizada no sistema embarcado, onde esta é responsável pela recepção das imagens, porém para utilizar essa biblioteca no Java é necessário incluir mais uma biblioteca, esta é o JNA, *Java Native Access*, que permite o acesso a bibliotecas nativas do sistema. Com as duas bibliotecas inclusas, é necessário que a configuração do codec do vídeo na IMC, seja a mesma configurada no sistema embarcado. Para facilitar para o usuário final todas as informações de erro, são informadas ao usuário através do campo de registro de *log* presente na IMC ou em uma janela de aviso.

4.4.1 CASOS DE USO

Nesta subseção será apresentado o diagrama UML de casos de usos. Foram identificados os seguintes atores:

- Usuário.
- Interface de Rede da IMC.
- Interface de Rede do SE.
- Motor Esquerdo e Direito.
- Servo Motor 1 e 2.

E os seguintes casos de uso:

- Autenticar Usuário.
- Alterar Credenciais.
- Envia Comando de Deslocamento.
- Envia Comando de Posicionamento da Câmera.

Na figura 27 está ilustrado os atores e os casos de uso identificados no projeto.

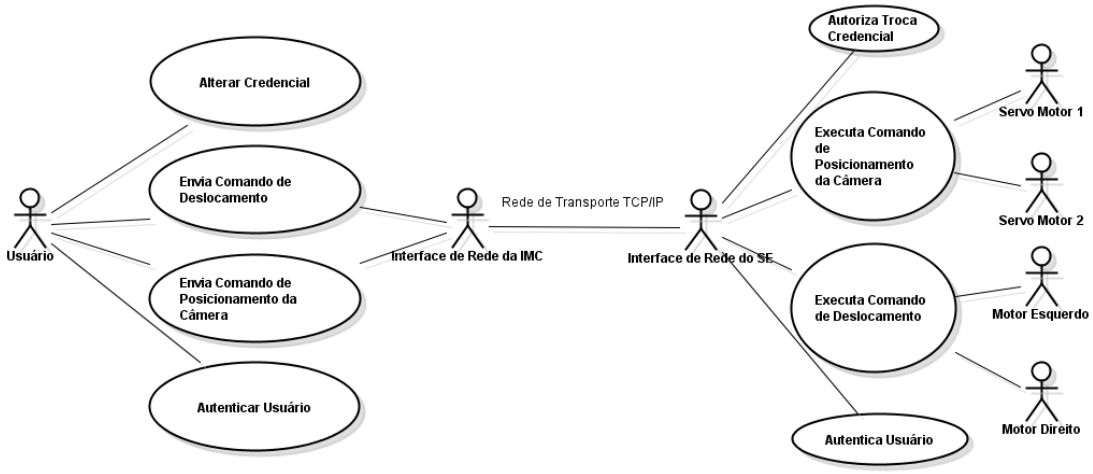

Figura 27: Diagrama de Casos de Uso (Autoria Própria)

4.4.1.1 ESPECIFICAÇÃO DOS CASOS DE USO

Nesta subseção será especificado detalhadamente cada caso de uso, esta especificação pode ser vista nos quadros 6, 7, 8 e 9.

Quadro 6: Caso de uso: Autenticar Usuário

Nome do caso de uso	Autenticar Usuário
Descrição	Autenticação do Usuário para comandar o robô
Ator principal	Usuário
Pré-condições	O usuário deve existir, e conexão deve estar funcionando
Pós-condições	O usuário está autorizado a enviar comandos
Fluxo básico	1. O usuário digita a senha
	2. A senha é enviada ao robô
	3. O robô verifica se a senha é a correta, e responde para IMC
	4. A IMC informa que o usuário está autorizado a enviar comandos
Fluxo alternativo	Se a credencial for inválida, a IMC solicita novamente a credencial

Quadro 7: Caso de uso: Alterar Credencial

Nome do caso de uso	Alterar Credencias
Descrição	Etapas percorridas para alterar a credencial
Ator principal	Usuário
Pré-condições	O usuário deve estar autenticado, e a conexão funcionando
Pós-condições	A credencial do usuário é alterada
Fluxo básico	1. O usuário solicita alterar a senha
	2. O usuário digita a senha atual
	3. O usuário digita a nova senha
	4. A IMC envia os pacotes para o robô
	5. O robô verifica se a senha está correta, e altera a nova senha
	6. O robô retorna para IMC, e esta informa que a senha foi alterada
Fluxo alternativo	Se a credencial for inválida, a IMC cancela a alteração

Quadro 8: Caso de uso: Envia Comando de Deslocamento

Nome do caso de uso	Envia Comando de Deslocamento
Descrição	Etapas percorridas para enviar comandos de deslocamento ao robô
Ator principal	Usuário
Pré-condições	A comunicação deve estar funcionando e o usuário autenticado
Pós-condições	O robô executa a ação enviada pelo usuário
Fluxo básico	1. O usuário solicita o comando de deslocamento
	2. O comando é enviado via interface de rede do usuário
	3. O comando é recebido na interface de rede do SE
	4. O robô executa a ação, via os dois motores
Fluxo alternativo	Se a comunicação não estiver ocorrendo o usuário é informado pela IMC

Quadro 9: Caso de uso: Envia Comando de Posicionamento da Câmera

Nome do caso de uso	Envia Comando de Posicionamento da Câmera
Descrição	Etapas percorridas para enviar comandos de posicionamento da câmera
Ator principal	Usuário
Pré-condições	A comunicação deve estar funcionando e o usuário autenticado
Pós-condições	O robô executa a ação enviada pelo usuário
Fluxo básico	1. O usuário solicita o comando de posicionamento
	2. O comando é enviado via interface de rede do usuário
	3. O comando é recebido na interface de rede do SE
	4. O robõ executa a ação, via os dois servos motores
Fluxo alternativo	Se a comunicação não estiver ocorrendo o usuário é informado pela IMC

4.5 RESULTADOS E TESTES

Após a finalização do desenvolvimento, a próxima etapa foi testar o sistema como um todo. Tendo em vista que todas as partes eram testadas antes de passar para a próxima. Nestes testes parciais eram sempre verificados se o objetivo daquela tarefa está de acordo, por exemplo se a comunicação TCP estivesse funcionando, ou seja as trocas de pacotes estavam ocorrendo corretamente. Consequetemente, foi possível passar para o próxima tarefa, analogamente, o mesmo procedimento foi realizado para as tarefas remanescentes.

Os testes foram realizados utilizando um roteador, configurado conforme mostrado na figura 8, os resultados dos testes e seus passos para validação são mostrados nos quadro 10 e 11. O resultado dos testes foi satisfatório, pois todos os objetivos e requisitos foram alcançados, porém foi visto que com a utilização, e assim o descarregamento da bateria do SE, é possível notar que, devido ao alto consumo de corrente, o sistema parava de responder, sendo assim necessário reiniciar o mesmo e carregar a bateria para uma total utilização.

A figura 28 apresenta uma imagem de frente de como ficou o robô em sua forma final e figura 29 apresenta a parte traseira do robô, foi utilizado de apoio aos componentes um acrílico, pois este é um material relativamente de fácil manuseio e que suportava todos os pesos.

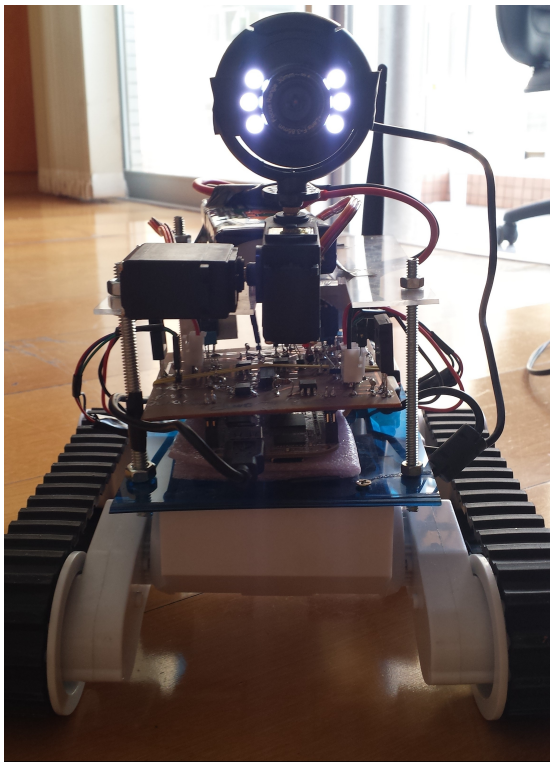

Figura 28: Imagem Final Robô Frente (Autoria Própria)

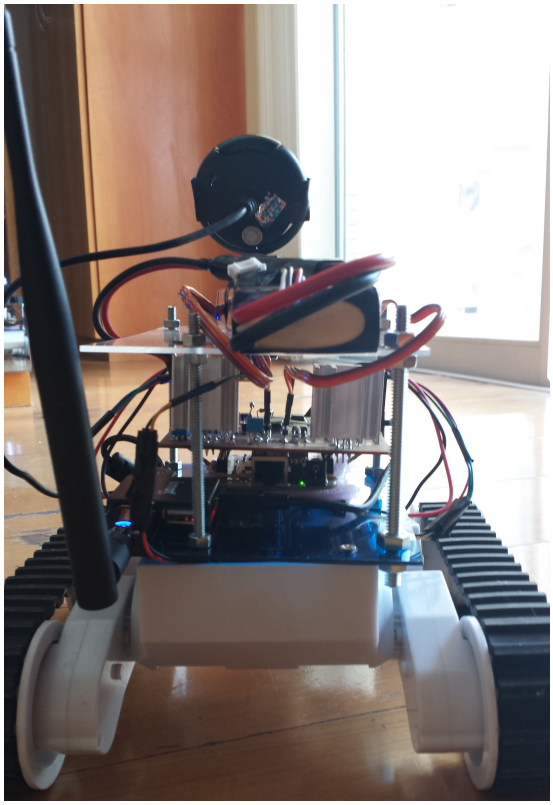

Figura 29: Imagem Final Robô Traseira (Autoria Própria)

A figura 30 apresenta como ficou a interface gráfica da IMC, onde é possível ver o campo onde é exibido o vídeo do ambiente monitorado, os botões de deslocamento, frente, direita e esquerda, botões de rotação da câmera com os ângulos atuais, uma campo de texto de *log*, onde o usuário pode analisar o histórico e o campo de configuração onde é apresentado as configurações do robô.

Figura 30: Imagem Final IMC (Autoria Própria)

Quadro 10: Testes de Validação do Sistema Embarcado

Requisito	Passos	Resultado
RF1	Foram enviados todos os comandos possíveis, através da IMC, e era analisado se a sua execução esteja dentro do esperado.	Todos os comandos foram executados.
RF2	Foi verificado se, após decifrar o pacote, este condizia com o esperado.	Todos os pacotes estavam dentro do padrão estabelecido.
RF3	Foi verificado, através do wireshark, se o pacote enviado a IMC, estava ou não cifrados.	Todos os pacotes analisados estavam cifrados.
RF4	Foi realizada a solicitação do vídeo através da IMC, e nela era verificado se a imagem recebida condizia com o que estava acontecendo realmente no ambiente e caso solicitado a compressão do vídeo se estava mesmo ocorrendo.	As imagens representam o que está ocorrendo no ambiente, e foi verificado que o vídeo era comprimido quando solicitado.
RF5	Foi enviado o comando de deslocar para frente através da IMC, e com isso era analisado se o robô conseguia executar seu movimento em linha reta.	Foi possível notar que o robô inicialmente não deslocava em linha reta, mas com o controle PID atuando este consegue deslocar em linha reta.
RF6	Foi enviado o pacote pedindo a solicitação de autenticação, inicialmente com a senha correta e depois com senha errada.	Os pacotes que chegavam com a senha correta o usuário era autenticado, já pacotes com senha errada o *software* não autoriza a autenticação e era enviado um pacote informando o erro.
RF7	Foi solicitado a alteração de senha, inicialmente a senha atual era digitada corretamente depois com senha errada, e depois enviado a nova senha.	Os pacotes com a senha correta eram trocadas a nova senha, os pacotes com senha errada a troca de senha não era autorizada.
RF8	Foi solicitado mais de uma conexão com o robô.	Todas as conexões tentadas eram rejeitadas pelo robô, sendo possível apenas a conexão quando a outra era fechada.

Quadro 11: Testes de Validação da IMC

Requisito	Passos	Resultado
RF1	Foram enviados todos os comandos possíveis para o robô, e era analisado se a sua execução esteja dentro do esperado.	Todos os comandos foram executados.
RF2	Foi verificado, através do wireshark, se o pacote enviado para o robô, estava ou não cifrados	Todos os pacotes analisados estavam cifrados.
RF3	Foi verificado se após decifrar o pacote se este condizia com o esperado	Todos os pacotes estavam dentro do padrão estabelecido
RF4	Foi solicitado o vídeo, e era verificado em uma janela, se a imagem recebida condizia com o que estava acontecendo realmente no ambiente, e se o vídeo estava comprimido, se este tivesse sido solicitado.	As imagens representam o que está ocorrendo no ambiente, e quando foi solicitada a compressão, esta estava ocorrendo.
RF5	Foi desligado o robô, e verificado se alguma mensagem aparecia informando que a conexão caiu.	Todos os testes de desconexão realizados apresentaram uma mensagem informando sobre a queda da conexão.
RF6	Foi clicado no botões de movimentação do robô e verificado se era possível continuar enviando os comandos enquanto estes estavam sendo executados.	Foram clicados em todos os botões e em todas as configurações de movimento em frente e em curva, e nenhum era habilitado enquanto o movimento não era terminado.
RF7	Foi solicitado a autenticação de senha, inicialmente a senha atual era digitada corretamente e depois com senha errada.	Os pacotes com a senha correta eram autenticados o usuário, os pacotes com senha errada eram informado ao usário que a senha não está correta.
RF8	Foi solicitado a alteração de senha, inicialmente a senha atual era digitada corretamente depois com a senha errada, e depois enviado a nova senha.	Os pacotes com a senha correta era trocada para , a nova senha, os pacotes com senha errada a troca de senha não era autorizada e informado ao usuário.
RF9	Foram enviados todos os pacotes possíveis, derrubado a conexão e conectado novamente.	Todas as informações enviadas para o robô foram mostrados na janela de *log*.

5 GESTÃO

Ao iniciar o projeto, foi definido um cronograma, para verificar se o projeto era viável e definir os tempo de cada tarefa a ser executada. Esse cronograma pode ser visto no quadro 12, onde a coluna Tarefa representa as tarefas a serem executadas, e a coluna Duração representa o tempo, em horas, que cada tarefas iria levar para ser finalizadas, o período de realização do projeto foi de 8 meses. Ainda no planejamento, foi realizado a previsão dos custos, demonstrado no quadro 13, com isso é possível verificar se o custo era o esperado pela disciplina.

Quadro 12: Duração das Tarefas do Projeto Planejada

Tarefa	Duração
Modelar o Projeto	50
Elaborar a Monografia de TCC	360
Adquirir os Componentes	10
Realizar Testes de Validação	30
Estabelecer a Comunicação TCP	100
Transportar o Vídeo UDP	100
Interfacear os Servo Motores	30
Interfacear os Motores DC e Implementar o Controle	40
Realizar o codec do Vídeo	100
Realizar a Autenticação do Usuário	100
Total	920

Quadro 13: Custo Planejado

Recursos	Preço em Reais
Placa BeagleBone	180
Chassis Rover 5	115
Antena 802.11g USB interface	31
2 Servos Motores	30
Câmera	40
USB-port hub	14
Componentes Elétricos	40
Baterias	170
Custo Total	620

Após o termino do projeto é possível realizar a comparação do planejado com o real, o quadro 14, apresenta qual foi o tempo total real utilizado na realização do projeto. Pode-se analisar que há uma diferença de 140 horas, isto ocorreu pois, ocorreram problemas com a realização do codec de vídeo, pois havia se planejado utilizar o algoritmo de H.264, porém com resultados não satisfatórios foram preciso realizar mais teste, para que a escolha do algoritmo fosse a melhor possível. Também ocorreram problemas com o interfaceamento dos servo motores, pois inicialmente os mesmos não estavam desacoplados, e ligados com a fonte externa, e ao ser conectados com a bateria, os servos apresentaram um ruídos, o qual desligava a BeagleBone, e para solucionar o problema os servo motores foram desacoplados.

Quadro 14: Duração das Tarefas do Projeto Final

Tarefa	Duração
Modelar o Projeto	50
Elaborar a Monografia de TCC	380
Adquirir os Componentes	40
Realizar Testes de Validação	80
Estabelecer a Comunicação TCP	40
Transportar o Vídeo UDP	60
Interfacear os Servo Motores	60
Interfacear os Motores DC e Implementar o Controle	60
Realizar o codec do Vídeo	170
Realizar a Autenticação do Usuário	120
Total	1060

Também é possível analisar na 15, que ocorreu uma diferença entre o custo do projeto planejado e o real em 110 reais de economia, a maior diferença está na bateria, pois foi possível emprestá-la. O único custo que aumentou foi a parte dos componentes elétricos, devido não ter sido planejado a construção da placa PCB. O planejamento inicial era utilizar um *shield* da

própria placa BeagleBone, porém com o grande aumento do número de componentes elétricos, teve que ser realizado uma placa PCB, e com isso o aumento no custo do projeto final, pois com o *shield* já construído não foi possível reaproveitar todos os componentes.

Quadro 15: Custo Final

Recursos	**Preço em Reais**
Placa BeagleBone	180
Chassis Rover 5	115
Antena 802.11g USB interface	31
2 Servos Motores	30
Câmera	40
USB-port hub	14
Componentes Elétricos	90
Baterias	10
Custo Total	510

6 CONCLUSÃO

Para a realização do projeto foram definidos inicialmente que eram necessárias 920 horas para a conclusão, porém ao final do projeto foram gastos 1060 horas, isto deve se ao fato que a escolha inicial do algoritmo que iria realizar o codec não deve resultados satisfatórios e com isso foi realizados testes com outros até que o resultado estivesse dentro do esperado para monitorar o ambiente. Foi necessário elaborar e construir uma PCB, tendo em vista projeto inicial prever a utilização uma placa *shield* do próprio microcontrolador, entretanto foi necessário adicionar fotoacopladores para os servos já que estes estavam gerando muito ruído no circuito. O custo do projeto também deve seu valor não condizente com o que foi previsto, porém para um valor menor, era inicialmente planejado gastar R$620,00 com o projeto, porém foi possível realizar o empréstimo da bateria, isto fez com tivesse uma economia de R$ 160,00 com a bateria, mas foi necessário realizar mais gastos com a parte de *hardware* R$50,00, com isso o custo final acabou sendo de R$510,00, uma economia de R$110,00 com o previsto.

Com o objetivo geral de realizar um sistema de monitoramento, inicialmente foi pensado na mobilidade que o robô teria que apresentar, com isso foi estudado tecnologias para transmissão de dados sem fio, dentre as opções foi escolhida a 802.11, que apresenta um bom alcance e uma taxa alta de transmissão. A taxa era uma preocupação, pois como seria realizado uma transmissão de vídeo, teria que ter uma taxa suficiente para que o usuário final não percebesse grandes perdas. Por isso além da preocupação com o meio de transporte, foi realizado uma compressão do vídeo, o algoritmo escolhido inicialmente o H.264, pois este apresenta uma maior eficácia de compreensão. Devido a dificuldade de conseguir uma configuração, que apresente uma qualidade de vídeo satisfatória, foram feitos testes com outros padrões de configuração e comparação com o algoritmo MPEG-4. Estes testes levaram em consideração, o processamento da placa, a memória utilizada, banda consumida e resultado final do vídeo. A melhor opção entre os algoritmos e configurações foi o MPEG-4. Além disso o *software* preserva o modo sem compressão, o qual consome mais banda mais preserva melhor qualidade, portanto possibilitando ao usuário o uso de compressão ou não, dependendo da necessidade do usuário.

Após definido o meio de comunicação, foi analisado as melhores formas de enviar os pacotes. Foi definido UDP, para o transporte de vídeo, e TCP para transporte de dados, pois este apresenta um controle de erros e a ordem de chegada. Como são utilizados quatro tipos diferentes de dados, comandos, autenticação, troca de senha e chave de sessão, cada um tem uma semântica diferente para ser possível diferenciar cada um. Definido o protocolo de comunicação, era preciso analisar qual a melhor escolha do microcontrolador, por isso foi definido utilizar a BeagleBone, pois esta permite utilizar um Linux embarcado, e com isso a facilidade de apresentar a pilha TCP/IP implementada, o acesso aos pino de *input*, *output* e PWM, através de *file descriptor*, e a facilidade de instalação dos *drivers* da antena, que realiza a comunicação através do protocolo 802.11, e com a da câmera de vídeo, utilizada para capturar vídeo do ambiente. Após esta etapa, foi necessário realizar os interfaceamentos dos componentes, inicialmente apenas os motores DC estavam com fotoacopladores, entretanto devido a problemas com ruído fez-se necessário realizar desacoplamento dos servo motores, consequentemente gerando mais custo ao projeto.

O *software* que executa no Linux embarcado foi escrito em Python e utiliza a biblioteca GSTreamer, responsável por realizar, a captura do vídeo da câmera, transporte UDP e o codec do video. Esta biblioteca é também utilizada na parte da IMC, onde o usuário comanda e monitora o robô. A IMC foi escrita em Java, é através desta que o usuário conecta-se ao robô, o controla e monitora o ambiente. Para realizar a conexão, é preciso que o usuário digite uma senha, isto serve para o que o usuário seja autenticado. Além da autenticação é utilizado o algoritmo de Cesar para garantir a confidencialidade dos pacotes de dados. Após a finalização do projeto foram realizados testes baseados nos requisitos previamente estabelecidos. Através destes foi possível verificar que todos os objetivos do projeto foram alcançados e que é possível monitorar o ambiente de forma intuitiva. O único empecilho encontrado foi o alto consumo de corrento do sistema, devido os seus periféricos, resulta em uma autonomia de aproximadamente duas horas.

Apesar do projeto cumprir todos os objetivos, ainda muito o que possa ser feito para melhorar o monitoramento. Inicialmente é aumentar o tempo de autonomia do robô, fazer com que robô possa percorrer rotas pré definidas. Além disso pode se acrescentar alguns sensores, por exemplo de fumaça, caso algo esteja pegando fogo é possível avisar o usuário, realizar um processamento de imagens, também seria interessante que o usuário pudesse definir quais sons seriam tocados, caso este visse a necessidade. É importante ressaltar que todas essas mudanças poderão ser realizadas, tendo em vista que o sistema é modular. Essas seriam apenas melhorias no projeto, pois o objetivo de monitorar o ambiente foi realizado.

REFERÊNCIAS

AMAZON.COM, C. **Rovio Robot Price**. 2013. Disponível em: <http://www.amazon.com/WowWee-Rovio-Enabled-Robotic-WebCam/dp/B001CQLGD6>. Acesso em: 31 de julho de 2013.

AMAZON.COM, C. **Spykee Spy Robot Price**. 2013. Disponível em: <http://www.amazon.com/Meccano-Erector-Spykee-Spy-Robot/dp/B000PA71U2>. Acesso em: 31 de julho de 2013.

ANG, K. H.; CHONG, G.; LI, Y. Pid control system analysis, design, and technology. **Control Systems Technology, IEEE Transactions on**, v. 13, n. 4, p. 559–576, 2005. ISSN 1063-6536.

BEACH, A. **Real World Video Compression**. 1. ed. Berkeley: Peachpit Press, 2008. ISBN 0321514696.

CISCO, S. **Enterprise Mobility 4.1 Design Guide**. San Jose: Cisco Systems, Inc., 2008.

COLEY, G. **BeagleBone System Reference Manual**. San Francisco, 2012.

DEAN, S. **Current and Future Trends in Medical Electronics**. [S.l.]: Texas Instruments, 2009.

FULLERTON, E.; SALES, C. **The History of Video Surveillance**. [S.l.]: Milestone Systems, 2008.

GOLSTON, J. Comparing media codecs for video content. In: **Embedded Systems Conference**. San Francisco: [s.n.], 2004. v. 250.

GUPTA, G.; MUKHOPADHYAY, S.; FRENCH, J. R. Wireless communications and control module of a web-enabled robot for distributed sensing applications. In: **Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE**. [S.l.: s.n.], 2008. p. 393–398. ISSN 1091-5281.

IEEE, WORKING GROUP. **(802.11) Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications**. New York, Março 2007.

JUNIOR, A. de O.; ALENCAR, R. A. da C. **Sistema de Indicadores de Percepção Social de Segurança Pública**. 2. ed. Brasília, Julho 2012.

KEMPER, F. **Relatório de Mercado de Segurança no Brasil 2012**. [S.l.], 2012.

MICROSOFT. **Network Bandwidth Requirements for Media Traffic**. [S.l.], Agosto 2013.

NOERGAARD, T. **Embedded System Architecture**. 2. ed. Oxford: Apress, 2005. ISBN 0123821967.

PAPADIMITRIOU, G. et al. **Wireless Network**. 1. ed. New York: John Wiley & Sons, 2002. ISBN 0470845295.

PETROU, M.; BOSDOGIANI, P. **Image Processing**. 1. ed. New York: John Wiley & Sons, 1999. ISBN 047074586.

RICHARDSON, I. **H.264 and MPEG-4 Video Compression**: Video coding for next-generation multimedia. 1. ed. New York: John Wiley & Sons, 2003. ISBN 0470848375.

RICHARDSON, I. **The H.264 Advanced Video Compression Standard**. 2. ed. New York: John Wiley & Sons, 2010. ISBN 0470516925.

ROBOTICS, P.; ELECTRONICS. **ROVER 5 Datasheet**. 2013.

SALLY, G. **Pro Linux Embedded Systems**. 1. ed. New York: Apress, 2009. ISBN 1430272279.

SAYOOD, K. **Introduction to Data Compression**. 3. ed. San Francisco: Elsevier, 2006. ISBN 012620862.

SPYKEEWORLD.COM, C. **Spykee Spy Robot**. 2013. Disponível em: <http://www.spykeeworld.com/spykee/US/index.html>. Acesso em: 31 de julho de 2013.

SUGISAKA, M.; HAZRY, D. User interface in web based communication for internet robot control. In: **ICCAS**. [S.l.: s.n.], 2005.

TANENBAUM, A.; WETHERALL, D. **Computer Networks**. 5. ed. Boston: Prentice Hall, 2010. ISBN 0132126958.

TELSTRA, G. H. Tcp performance. **The Internet Protocol**, Cisco Systems, v. 3, No 2, p. 1, 2000.

WOWWEE.COM, C. **Rovio Robot**. 2013. Disponível em: <http://www.wowwee.com/en/products/tech/telepresence/rovio/rovio>. Acesso em: 31 de julho de 2013.