UNIVERSIDADE ESTADUAL DO PIAUÍ - UESPI
CAMPUS PROF. ALEXANDRE ALVES DE OLIVEIRA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARLO ZENI BRAGA DE ARAUJO

PROJETO CONTROLE: PROTÓTIPO DE TELEROBÓTICA USANDO PYTHON E ARDUINO

PARNAÍBA
2013

MARLO ZENI BRAGA DE ARAUJO

PROJETO CONTROLE: PROTÓTIPO DE TELEROBÓTICA USANDO PYTHON E ARDUINO

Monografia submetida ao Curso de Bacharelado em Ciências da Computação da Universidade Estadual do Piauí, como parte dos requisitos para a obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. Francisco das Chagas Rocha

PARNAÍBA
2013

A658p

Araujo, Marlo Zeni Braga de
Projeto controle: protótipo de telerobótica usando python e arduino/Marlo Zeni Braga de Araujo.- Parnaíba: UESPI,2013.
61f. : il.

Orientador: Msc. Francisco das Chagas Rocha
Monografia (Graduação em Ciência da Computação) – Universidade Estadual do Piauí, 2013.

1. Telerobótica2. Python 3. Arduino4. Wi-fi5. Hardware livreI.Rochà, Francisco II. Universidade Estadual doPiauí III. Título

CDD 005.133

MARLO ZENI BRAGA DE ARAÚJO

SOLUÇÕES SIMPLES PARA TELEROBÓTICA UTILIZANDO PYTHON E ARDUÍNO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual do Piauí – UESPI, Campus Prof. Alexandre Alves de Oliveira, como parte das exigências da disciplina de Estágio Supervisionado, requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: M.Sc. Francisco das Chagas Rocha

Monografia Aprovada em: **16 de março de 2013.**

Banca Examinadora:

Prof. M.Sc. Francisco das Chagas Rocha
UESPI/Parnaíba – Orientador

Prof. M.Sc. Anderson Passos de Aragão
UESPI/Parnaíba – Avaliador Interno

Prof. M.Sc. Francisco Marcelino Almeida de Araújo
IFPI/Teresina – Avaliador Externo

DEDICATÓRIA

Dedico este trabalho aos que direto ou indiretamente, me incentivaram, e principalmente à minha amada namorada e futura esposa Nathalee Paloma.

AGRADECIMENTO

Agradeço aos professores Francisco Rocha pelas orientações no desenvolvimento deste trabalho, Francisco Marcelino por ter me proporcionado uma ampla visão sobre equipamentos de hardwares, bem como ter me incluído em seu grupo de desenvolvimento de tecnologias.

Agradeço ainda, todos os amigos não mencionados que, de alguma forma, me ajudaram para que eu pudesse realizar este trabalho.

EPÍGRAFE

"O mais competente não discute, domina a sua ciência e cala-se".

Voltaire

RESUMO

Este trabalho descreve um protótipo de Telerobótica e visa mostrar uma possível solução para problemas como explorar ambientes nocivos ao homem, proporcionando a execução de tarefas onde a vida do usuário não corra riscos. Para isto, foi abordado o uso da linguagem Python, trabalhada em conjunto com uma placa micro controlada, o Arduino. Serão mencionados mais detalhadamente os Diagramas de Classe, Casos de Uso, Diagrama de Atividades, além dos módulos do Python usados, e uma arquitetura de um protótipo robótico em formato de um veículo automotor de quatro rodas. Efetuaram-se testes que mostram o desempenho desta arquitetura e suas execuções foram analisadas. Foram encontrados alguns problemas que serão apresentados neste trabalho, bem como, suas possíveis soluções.

PALAVAS-CHAVES: Telerobótica. Python. Arduino. Wi-Fi. Hardware Livre.

ABSTRACT

This paper describes a solution for Telerobotic systems. Aims to show a solution to problems such as explore harmful environments to humans, providing task execution where the researcher's life do not take risks. For this, we will addressed the use of Python language, where it will work with a board of open source Arduino. Will be mentioned in more detail the Class diagrams, Use Case, Activity Diagram, besides the Python modules used, and an architecture design of a robot in the shape of a motor vehicle with four wheels. We conducted tests that show the performance of this architecture and their performances were analyzed. We found some issues that are presented in this work, as well as their possible solutions.

KEYWORDS: Telerobotic. Python. Arduino. Wi-Fi. Hardware Open Source.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O homem, desde os primórdios de sua existência, com sua sede de descobrir o novo, tinha o nível de suas explorações limitadas por seus conhecimentos pouco desenvolvidos ou não levados em consideração e/ou não estimulados. Mesmo antes de surgir o conceito de tecnologia, esta era desenvolvida lenta e gradativamente sem a percepção de seu criador. A partir da segunda guerra mundial (1939 a 1945), século XX, a tecnologia teve uma evolução suficiente para que fossem realizadas as utopias que o ser humano não imaginaria que pudessem tornar-se realidade. Para corroborar o exposto Ravetz (1971) apud Menéndez (2004) contribui ao afirmar que:

> A ciência tradicional, a ciência acadêmica, preocupava-se basicamente em conceber teorias verdadeiras seguindo as diretrizes marcadas, por exemplo, pelo método científico. Mas isso começou a se modificar depois da Segunda Guerra Mundial, devido ao processo de industrialização da ciência e a criação de projetos de investigação e desenvolvimento em grande escala, como por exemplo, o Projeto Manhattan para construir a bomba atômica, os programas espaciais, a engenharia genética, a realidade virtual, etc. Construía-se uma nova ciência que, segundo Echeverria (2001), era freqüentemente denominada tecnociência ou Big Science, a qual requeria grandes equipamentos e vultuosos recursos econômicos para ser desenvolvida.

Baseado nesse pressuposto, este trabalho surgiu da curiosidade de construir um protótipo de Telerobótica desenvolvido a partir do Python e Arduino. Essa temática causou a seguinte problemática: ausência de fontes teóricas que explanassem tal assunto a partir do Python e Arduino. Em resumo, este trabalho surgiu a partir de saber como funcionava um protótipo, usando o Python e Arduino. Para a execução da pesquisa pautou-se nos seguintes objetivos: como objetivo geral, sugerir um Protótipo de Telerobótica, usando o Python e Arduino; e como objetivos específicos, realizar a comunicação entre dois ou mais computadores através da rede de forma mais simples, usando Socket; transmitir imagens pela rede; obter comunicação serial entre o Python e o Arduino.

O Projeto Controle, um Protótipo de Telerobótica usando Python e Arduino visa realizar ações com eficiência e precisão, isso através de um sistema tanto autônomo quanto controlado por um usuário, independentemente da distância entre estes dois atores principais, ter em vista a segurança do controlador ao explorar lugares que, ainda nos dias

de hoje, a presença humana torna-se difícil, arriscada ou impossível de se permanecer por tempo necessário para que se realize um estudo que satisfaça a sede de conhecimento.

Esses lugares de difícil exploração como grandes profundidades de fendas no fundo no oceano, superfície de outros planetas, construções que correm riscos de desabamentos, desarmamento de bombas, manuseio de produtos radioativos, tarefas que devem ser executadas por um profissional específico, e não haver a possibilidade de que este esteja no local da tarefa, seja por qual motivo, ou com precisão necessária para obter-se resultado satisfatório, são situações que hoje são facilmente contornadas com o uso de equipamentos que integram algumas áreas da ciência como mecânica, eletrônica, informática, entre outras.

O estudo desses equipamentos foi dando o surgimento a uma nova área da tecnologia. Máquinas que fossem capazes de executar instruções de uma forma precisa, evitando erros previsíveis, não importando o grau de nocividade do ambiente atual para a saúde humana. A robótica surgiu como a ciência dos sistemas que interagem com o mundo real, com pouco ou mesmo nenhuma intervenção humana (MARTINS, 2006).

A robótica, como parte operacional, e a computação, como parte pensante, podem trabalhar em conjunto na resolução destes problemas. A interação se dá da seguinte forma: um software que tenha duas versões, uma versão para enviar comandos e a outra para receber comandos, sendo que, esses comandos devem originar-se de dispositivos de entrada como teclado, joystick, entre outros, para que seja possível a interação de um usuário comunicando-se com um robô através de uma rede sem fio onde este deve ser capaz de realizar movimentos dependentes destes comandos.

Além de comandos enviados a um robô, a Telerobótica, ciência que controla robôs a distância, usando sobretudo, conexões sem fio, tem uma característica específica. No ambiente em que o robô se encontra, deve haver uma câmera, onde todos os movimentos do robô devem ser monitorados através desta. O robô deve ser capaz de capturar imagens da câmera e enviá-las de volta ao usuário, mantendo essa comunicação em todo o período de execução, possibilitando assim que o homem não tenha contato com o ambiente arriscado e perigoso.

Através de testes realizados, propõe-se uma metodologia para Telerobótica utilizando a arquitetura Cliente/Servidor implementada com base em *socket*[1] desenvolvida na linguagem Python para plataforma desktop, permitindo assim, a troca de informações

[1] Programação voltada à rede de computadores.

entre dispositivos conectados a uma rede. A linguagem Python permite a comunicação com dispositivos de entrada através da biblioteca *Pygame*, que dá suporte também manipulação da webcam e dispõe de bibliotecas para lidar com interface com o usuário, além de se comportar muito bem com comunicação serial, possibilitando a interação com placas micro controladas ou hardwares externos como o Arduino.

O Arduino é um hardware livre, usado principalmente em projetos de automação, robôs, além disso é uma plataforma de circuito com um microchip programável, que dispõe de placas e/ou dispositivos adicionais para a realização de tarefas diversas como sensor de distância, aceleração, umidade, velocidade, conexões de rede com fio ou sem fio, Bluetooth, Wi-Fi entre outros. Nesse caso, o Arduino deve ser integrado com um controlador de motores de corrente contínua, tendo em vista que ele não é capaz de controlar, por si só, pulsos elétricos de voltagens muito elevadas. Sua alimentação é baseado entre 3,7 volts e 9 volts com 500 mA (miliampéres) que é a corrente fornecida pela porta USB 2.0 de um computador. Mas é necessário manipular grandes voltagens o suficiente para alimentar os motores responsáveis pela locomoção do robô, sendo necessária, assim, uma placa adicional que, integrada ao hardware, possibilite o controle de voltagens altas o suficiente para não danificá-lo, essas placas são os Drivers.

Integrando-se Python com Arduino, as possibilidades de desenvolver um robô são muitas. A linguagem de programação, Python seria responsável pela programação que exige mais recursos como a comunicação através da rede com o módulo *socket*, já contido em sua biblioteca padrão permitindo a comunicação. Para tal comunicação, faz-se necessário ter um computador acoplado na estrutura robótica com uma câmera, onde o Pygame dá suporte à captura de imagens e as envia para o usuário. Já o Arduino se responsabiliza pela interface entre o Python e os motores, que por fim, realizam os movimentos do robô. Dessa maneira, pode-se enxergar uma possível solução para problemas como: exploração de ambientes hostis, desarmamento de bombas, dentre outros já mencionados.

Esse trabalho é composto por cinco capítulos com o objetivo de proporcionar uma melhor visão sobre a solução proposta. O mesmo divide-se em:

- Referencial Teórico, onde será melhor abordado sobre Telerobótica com alguns trabalhos relacionados, além de incluir um pouco da história do Python e seus módulos usados para se desenvolver o programa proposto, e ainda uma visão geral do equipamento que se analisou ser necessário para o desenvolvimento da estrutura Telerobótica;

- A Modelagem UML do sistema será abordada no capítulo três, com explanações e representações gráficas dos diagramas de Casos de Uso, diagrama de Classes e diagrama de Atividades que se fizeram primordiais para que se pudesse proporcionar um melhor entendimento do projeto;
- Estrutura e Arquitetura que serão explicadas no capítulo quatro, apontará a maneira que o referencial teórico foi organizado para se estruturar a proposta desse trabalho;
- Considerações Finais virão com o objetivo de realizar uma análise geral da arquitetura, resultados alcançados com a estrutura obtida até agora, dificuldades encontradas no desenvolvimento e trabalhos que poderão ser desenvolvidos a partir da proposta apresentada.

2 REFERENCIAL TEÓRICO

2.1 TELEROBÓTICA

Sabe-se que as transformações sociais e consequentemente tecnológicas não ocorrem repentinamente, mas sim através de acontecimentos gradativos do cotidiano que mesclados a outros acontecimentos que vão surgindo, geram as grandes descobertas da humanidade. Uma exemplificação disso está na reformulação de teorias falhas e imprecisas que a partir de novos experimentos, são aprimoradas. Costa (2002) expõe que:

> Todos reconhecemos que inovações tecnológicas dos mais variados tipos introduzem transformações em nossas vidas. Além das transformações que presenciamos em primeira mão, somos capazes de ter acesso a inúmeras outras quando estabelecemos contato, por meio de relatos dos mais velhos, livros, filmes, viagens, etc., com os modos de vida de épocas e lugares em que uma ou outra tecnologia ainda era desconhecida. Esse tipo de contato com o antes de determinada tecnologia, torna fácil perceber as transformações por ela geradas no depois.

Percebe-se, portanto, que muitos estudos antes de se solidificarem enquanto saber sistemático, passam também por sucessivas mudanças. Como é o caso de um dos focos deste trabalho, a Telerobótica. Nesse sentido, a Telerobótica, ou Telepresença, de certa forma surgiu a partir dos primeiros estudos sobre operações realizadas a distância sem a presença do operador humano, atrelado a necessidade de exploração de ambientes noviços ao homem, como superfície de outros planetas, escavações em lugares nocivos a desabamentos, profundezas do oceano, entre outros. Álvares (2012) expõe que:

> Inicialmente desenvolvida para a manipulação de materiais radioativos, a teleoperação permite que um operador exerça força e realize movimentos sobre uma máquina remota, e ainda receba realimentação sensorial, geralmente através de dados visuais, sonoros ou táteis. Com a introdução da tecnologia de teleoperação, foi possível o desenvolvimento de interfaces capazes de prover uma interação satisfatória entre homem e máquina, permitindo que serviços de grande destreza fossem realizados.

Sob essa perspectiva a Telerobótica apresenta-se como a ciência que alia a Robótica com a Telemática, sendo esta última na visão de Pinheiro (2005) a "área do conhecimento humano que reúne um conjunto e o produto da adequada combinação das tecnologias associadas à eletrônica, informática e telecomunicações", ou seja, a união da

Telemática com a Robótica engendra o ato de controlar um robô à distância, integrando assim, várias áreas do conhecimento citadas por Pinheiro. A Telerobótica nesse sentido apresenta uma importância ímpar na medida em que, possibilita às pessoas a realização de ações a longas distâncias. O acesso de pesquisadores a ambientes hostis como grandes profundidades de fendas no fundo no oceano, superfície de outros planetas, construções que correm riscos de desabamentos, desarmamento de bombas, manuseio de produtos radioativos, que podem oferecer algum risco a suas vidas, são solucionados a partir de equipamentos desenvolvidos para tal finalidade. Para realizar essa tarefa, é primordial um equipamento Telerobótico, principalmente em estudos onde são necessárias a resistência e precisão de um robô e o conhecimento de um especialista humano, promovendo a redução de risco de vida dos pesquisadores, uma vez que não seja necessário o operador do sistema e a máquina operada estarem no mesmo local.

> Desde seu início modesto em 1940, quando o primeiro Tele Operador foi projetado, o foco foi principalmente para aplicações a serem realizadas no espaço, materiais nucleares e operações subaquáticas até os anos 80 [...] Tele Cirurgia, Telerobótica semi-autônoma, manutenção de linha viva de energia, e outros. (CHOPRA *et al*, 2008).

Sua aplicação é diversificada, fornece a oportunidade de aumentar o conhecimento humano com explorações no espaço (Figura 1) onde astronautas podem utilizar-se da Telerobótica para explorar outros planetas a partir de suas atmosferas com redução de custos e risco às vidas dos exploradores. Do mesmo modo em que cientistas e engenheiros alcançam profundezas dos mares remotamente a partir de navios na superfície.

Figura 1 – Ano 2033, Humanos na órbita de marte com robôs na superfície
Fonte: http://telerobotics.gsfc.nasa.gov/

2.2 TRABALHOS RELACIONADOS

Além do espaço, a Telerobótica pode ser usada em distâncias menores, principalmente no que tange em salvamento de vidas. Os sistemas cirúrgicos Telerobóticos Zeus e da Vinci permitem a cirurgiões, a realização de cirurgias através de uma visão remota (BALLANTYNE, 2002).

Existem vários programas que possibilitam o controle de um robô através de uma rede, no Brasil pode ser achado um deles. O RobWebCam (Figura 2) desenvolvido pelo Grupo de Automação e Controle Departamento de Engenharia Mecânica - GRACO, da Faculdade de Tecnologia da UNB, é um sistema composto por um manipulador com 2 graus de liberdade, que significa as capacidades de movimento em um plano/ambiente tridimensional, câmera de vídeo, conexão com a rede Internet, computadores e os drivers de comunicação entre os diversos componentes (GRACO, 1999). Seus desenvolvedores disponibilizam um site para teste desse sistema por 30 segundos, acessível em http://www.graco.unb.br/robwebcam.html.

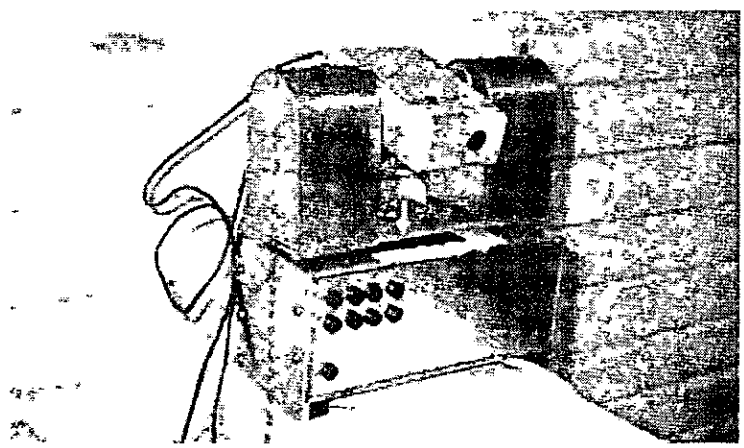

Figura 2 - RobWebCam
Fonte: http://www.graco.unb.br/robwebcam.html

Outro trabalho relacionado que pode ser citado nesta pesquisa é o *Curiosity* (Figura 3), veículo mecânico, construído para estar presente em ambientes hostis, como por exemplo a superfície de outro planeta. Apesar do *Curiosity* ter sido construído a partir de estudos diferentes dos realizados neste trabalho, é possível afirmar que o Protótipo de Telerobótica, denominado nesta pesquisa de Projeto Controle, pode alcançar objetivos semelhantes aos do veículo, na medida em que foi projetado para frequentar ambientes de difícil acesso humano.

> O MSL (Mars Science Laboratory), sonda especial lançada no dia 26 de novembro de 2011, pela Nasa. Dentro dessa sonda foi levado o *Curiosity*, um veículo mecânico para a exploração do solo marciano. O equipamento espacial posou em Marte no dia 6 de agosto de 2012, mais precisamente na cratera Gale, após viajar no foguete Atlas V.[...] O curiosity utiliza câmeras de navegação batizadas de Navcams, equipamento instalado num mastro de metal com a possibilidade de captar imagem em 360°. O local de trabalho do novo robô fica nas redondezas da Cratera Gale ao sul do equador marciano, região do tamanho dos estados americanos de Connecticut e Rhode Island juntos. (REBOUÇAS, 2012)

Desenvolvido para realizar pesquisas em ambientes, em que a presença humana, nos dias de hoje ainda se torna impossível, o Curiosity faz parte dos avanços tecnológicos da sociedade contemporânea.

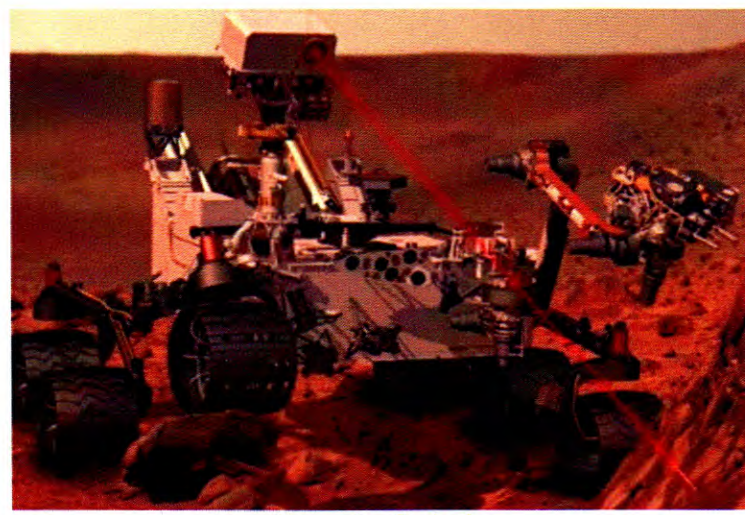

Figura 3 - Curiosity
Fonte: http://info.abril.com.br/noticias/ciencia/curiosity-pode-ter-feito-descoberta-historica-em-marte-21112012-42.shl

2.3 LINGUAGEM PYTHON

Para o melhor entendimento do trabalho será abordada a história da linguagem de programação Python com o intuito de saber em que condições sociais e, sobretudo, tecnológicas, esta surgiu. Em seu site[2] oficial, pode-se encontrar uma parte, como resumo, da história dessa linguagem revolucionária.

Sucessor de uma linguagem chamada ABC e tendo como principal autor Guido van Rossum, por volta de 1990, foi iniciado o desenvolvimento de Python na Holanda, onde evoluiu até a versão 1.2, em 1995, Guido, já em Reston, Viginia, realizou vários

[2] http://www.python.org/getit/releases/2.7/license/

lançamentos do software, mas só progrediu até a versão 1.6, no ano de 2000, com uma parceria com a *BeOpen.com*, Python atingiu sua versão 2.0, posteriormente, Guido e os demais desenvolvedores do Python uniram-se a *Digital Creations* e a partir do Python 2.1, e suas versões alfa e beta, toda propriedade intelectual é propriedade de uma organização sem fins lucrativos, *Python Software Foundation.*

O Python é definido como uma linguagem de alto nível e de código aberto, tornando-se de fácil compreensão e aprendizado, com estruturas de dados já implementadas, oferecendo um poder considerável ao programador com orientação a objeto além de ser muito familiar, parecendo com qualquer outra linguagem de programação. "Várias coisas. As respostas mais longas começam por afirmar que há muitas coisas que são familiares, também. Python é muito parecido com qualquer outra linguagem de programação de propósito geral, com as declarações, expressões, operadores, funções, módulos, métodos e classes" aponta Barry (2011), destacando semelhança do Python com as outras linguagens.

Python possui seu modo de execução em tempo real, ou seja, ao mesmo tempo em que é digitada no interpretador, é aceita na maioria das plataformas. O conteúdo de sua biblioteca padrão é extensa, baseada em outras linguagens sendo possível a integração com C/C++, ou qualquer linguagem acessível a partir de C, através da biblioteca *ctypes*, e JAVA pela biblioteca jython.

Para corroborar com o exposto, Python permite que você escreva o código necessário rapidamente, sendo necessárias poucas linhas de código para realizar algumas tarefas. E graças a um byte compilador altamente otimizado e bibliotecas de apoio, códigos de Python executam rápido o suficiente para a maioria das aplicações. (*PYTHON SOFTWARE FOUNDATION*, 2012).

> Assembly foi uma dádiva de Deus para aqueles que lutaram com código de máquina, em seguida veio Fortran, C e Pascal, que levou computação para outro nível e criou a indústria de desenvolvimento de software. Através da linguagem C vieram as mais modernas linguagens compiladas, C++ e Java. E ainda com sistemas poderosos e acessíveis, linguagens de script interpretadas como Tcl, Perl, e Python. (CHUN, 2009, p.6, traduzido pelo autor).

Segundo Chun (2009), entre as características do Python, destacam-se o alto nível ocasionado pelo processo de evolução das linguagens Assembly, FORTRAN, C,

Pascal, onde C originou linguagens de hoje como C++ e Java, e em um nível mais alto com poderosos sistemas de linguagens de script interpretadas como Tcl, Perl e Python.

> Python pode ser encontrado em uma ampla variedade de sistemas, contribuindo para seu crescimento rápido e contínuo no domínio da computação de hoje. Python é escrito em C, e por causa da portabilidade do C, Python está disponível em praticamente todo tipo de plataforma que tem um compilador ANSI C. Embora existam alguns módulos específicos de cada plataforma, geralmente qualquer aplicação escrita em Python em um sistema será executada com pouca ou nenhuma modificação na execução em outro sistema. (CHUN, 2009, p.6, traduzido pelo autor).

Observa-se que uma vantagem é sua portabilidade. Um programa feito em Python escrito em um sistema operacional qualquer, pode executar em outro sistema com pouca ou nenhuma modificação. Graças a sua base desenvolvida em C, Python está disponível em praticamente toda plataforma que possua um compilador C ANSI.

Chun ainda explana o poder que Python fornece ao programador de estar no total controle de erros ou exceções, dando informações do porque o programa deu erro e a localização do erro no código (nome do arquivo, número da linha, a função chamada, etc.), e ainda fornece uma capacidade de monitoração de erros em tempo de execução, possibilitando algum meio de evasão para esses erros. Além de Chun a classificar como uma linguagem byte-compilada, como quase sempre as linguagens interpretadas são mais lentas que as compiladas, pela execução não ser realizada em uma linguagem de sistema binário nativa, Python resulta em uma forma intermediária, aproximando-se da linguagem de máquina.

A linguagem Python é usada para criar tudo em aplicações WEB, programação de sistemas. O *NASA's Jet Propulsion Lab* (Laboratório de Propulsão à Jato da NASA), o Serviço Nacional de Meteorologia dos Estados Unidos, sistema de compartilhamentos de arquivos BitTorrent, Google e YouTube são seus usuários, além de sua popularidade no desenvolvimento de jogos como o jogo online EVE. A única restrição em relação ao Python são algumas funções de seus módulos que não estão disponíveis para todas as plataformas.

2.4 BIBLIOTECA PADRÃO DO PYTHON

O Python, ao ser instalado, leva consigo em sua biblioteca padrão, centenas de módulos que contêm ferramentas que permitem ao programador uma interação com o sistema operacional, interpretador e internet. Seu conteúdo extenso escrito em C, permite um melhor uso de funcionalidades do sistema bem como dispositivos de entrada e saída de dados. Os próximos tópicos explicarão o funcionamento dos módulos utilizados no desenvolvimento do programa proposto.

2.4.1 Time

O módulo *time* disponibiliza funções relacionadas ao tempo, mas como outros módulos, nem todas as funções são disponíveis para todas as plataformas. Foi utilizada a função *strftime*, que recebe uma *string* por parâmetro, especificando o formato da data retornada. Foi usada para nomear arquivos gerados pelo programa. O quadro abaixo apresenta as opções usadas de acordo com o desejado:

Quadro 1 – Strings e seus significados na função *strftime*

String	Significado
a	Nome do dia da semana abreviado
A	Nome do dia da semana completo
b	Nome do mês abreviado
B	Nome do mês completo
c	Representação de data e hora apropriada
d	Dia do mês como número decimal [01,31]
H	Hora (24 horas) como número decimal [00,23]
I	Hora (12 horas) como número decimal [01,12]
j	Dia do ano como número decimal [001,366]
m	Mês como número decimal [01,12]
M	Minuto como número decimal [00,59]
p	Equivalente a AM ou PM
S	Segundos como número decimal [00,61]
U	Número da semana do ano (domingo como primeiro dia da semana) como número decimal [00,53].
w	Dia da semana como número decimal [0,6]
W	Número da semana do ano (segunda como primeiro dia da semana) como número decimal [00,53].
x	Representação de data apropriada
X	Representação de tempo apropriada
y	Ano, sem casa de centena e de milhar, como número decimal [00,99]
Y	Ano com casa de centena e de milhar
Z	Nome da zona de horário

A Figura 4 mostra um exemplo de como a função *strftime* foi usada. Sendo necessário, primeiramente, importar-la módulo *time*. Por conseguinte, invocá-la passando por parâmetro a *string* '%Y-%m-%H-%M-%S' significando: ano com quatro dígitos, mês com dois dígitos, hora em decimal, minuto em decimal e segundos em decimal.Retornando assim os dados correspondentes a cada letra indicada: '2012-10-15-02-51'.

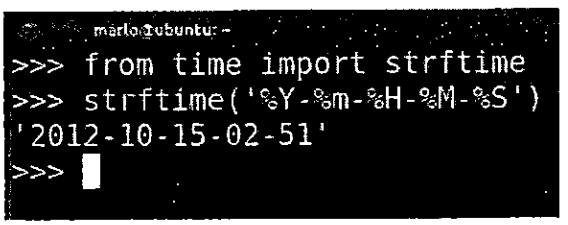

Figura 4 – Módulo *time*, função *strftime*

Além de acessar detalhes de tempo do sistema, a biblioteca *time* pode ser usada no momento em que o programa cria um arquivo por comando do usuário, neste trabalho, esta função foi usada para o fornecimento do nome dos arquivos. Para que dois arquivos não sejam criados com o mesmo nome, sendo que, o último arquivo criado sobrescreveria outro de mesmo nome, se seu nome for data e hora de criação (ano-dia-hora-minuto-segundo), a chance de sobrescrever outro arquivo existente é mínima.

2.4.2 Módulo *os*

O módulo *os* permite o uso de funcionalidades dependentes do sistema operacional. Ele contém todas as chamadas do sistema operacional habituais que se usa em programas em C e *shell scripts*. Suas chamadas lidam com diretórios, processos, variáveis do *shell*, e afins (Lutz, 2011, p.90, traduzido pelo autor). Foram usados o módulo *path* e as funções *getcwd* e *chdir* que são encontrados dentro do módulo *os*. *Path* é composto por algumas funções extremamente úteis relacionadas a caminhos de diretórios. Uma delas é a função *exists* que recebe uma *string* e retorna *True* (verdadeiro) se esta *string* representar um diretório válido ou existente, e *False* caso contrário.

A Figura 5 destaca um exemplo de uso do módulo *path*. Importando-o do módulo *os* e usando a função *exists*, passa-se por parâmetro a string '/home/marlo'. Retornando *true* a função indica que esse diretório existe.

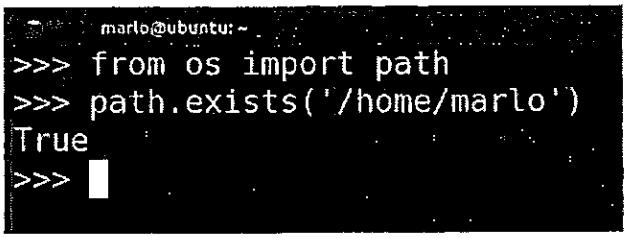

Figura 5 – Módulo *os*, módulo *path*

No desenvolvimento de um programa a função *getcwd* é muito útil quando se quer saber o diretório atual de execução. O software apresentado neste trabalho proporciona ao usuário a liberdade de escolher o diretório de salvamento de seus arquivos: imagens capturadas ou vídeos provenientes do servidor, que por sua vez, captura imagens de uma câmera e as envia ao usuário usando a rede como meio de comunicação. Mas para isso é necessário o auxílio de outra função abordada mais a frente.

A Figura 6 ilustra um exemplo de como a função *getcwd* pode ser usada. Após ser importada do módulo *os*, sendo chamada, esta função retorna o diretório atual de execução do interpretador do Python. No exemplo, o interpretador foi executado a partir do diretório '/home/marlo'.

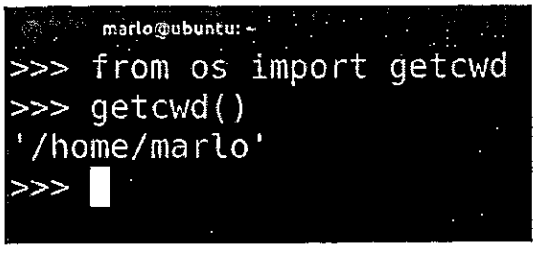

Figura 6 – Módulo *os*, função *getcwd*

O programa permite ao usuário mudar o diretório onde deseja salvar suas imagens ou vídeos capturados, a função *chdir* admite tal mudança, assim sendo, o diretório atual de execução passa a ser substituído por um diretório passado por parâmetro com *string*, mas para isso, faz-se necessário que a *string* represente um diretório válido, caso contrário o interpretador retornará um erro justificando que "O sistema não pode encontrar o arquivo especificado".

Proporcionando total liberdade ao usuário no momento em que for salvar seus arquivos, a Figura 7 mostra um exemplo de como o diretório de execução pode ser facilmente alterado usando a função *chdir*. Ao se executar a função *getcwd*, que mostra o diretório atual de execução do interpretador, antes e depois da função *chdir*, percebe-se a

praticidade oferecida, pelo programa proposto, ao usuário, reduzindo, também, o esforço do programador no desenvolvimento desse aplicativo.

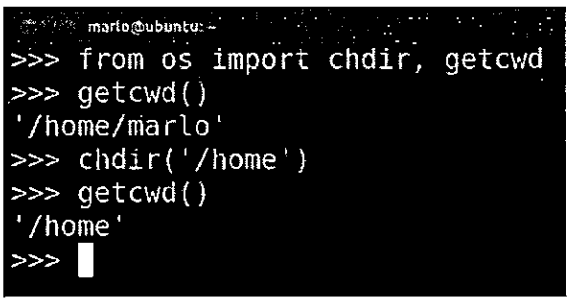

Figura 7 – Módulo os, funções *chdir* e *getcwd*

2.4.3 Módulo *Thread*

Os programas de computador de hoje em dia não funcionam somente com um processo principal, uma *thread* é uma tarefa criada por um programa principal, para fazer execuções concorrentes a ele, podendo realizar efeitos visuais para obter uma interface mais amigável com o usuário, ou realizar processamentos fora do bloco principal do processo inicial, executados sem que o usuário perceba. Uma *thread* comum e muito utilizada é a barra de status, criada para executar de forma concorrente ao programa principal carregando a proporção da execução do processo.

Python possui um módulo chamado *thread* que possui primitivas de baixo nível para trabalhar com *threads* dividindo um espaço de dados global, todos os *threads* são executados dentro do mesmo processo, ou seja, o processo pai. Uma das funções definidas nesse módulo é *start_new_thread* que recebe uma função ou método de uma classe e a executa a qualquer momento de execução do programa principal.

Na Figura 8, foi criada uma função *soma* que recebe dois dados quaisquer e que mostra na tela a soma se forem enviados números. No momento em que a função *start_new_thread* é executada, a função soma também é executada imprimindo o valor "7" como retorno da soma de "3" e "4" mesmo antes que o interpretador retorne a posição inicial de espera para um novo comando, mostrando, assim, que a função foi executada concorrentemente ao processo principal representado pelo interpretador.

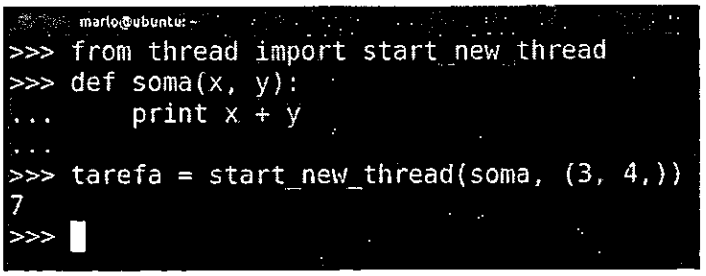

Figura8 – Módulo *thread*, função *start_new_thread*

2.4.4 Módulo *Threading*

Este módulo constrói interfaces de tarefas (*threads*) de nível mais alto, acima do módulo *thread* de nível mais baixo baseado em objetos e classes. Uma classe pode ser instanciada com o módulo *threading* e é transformada em uma tarefa. Dentro desse módulo está definida uma classe *threading. Thread* que representa uma *thread* de controle. A classe com instância de *threading. Thread* é executada no momento em que é chamada, ela deve conter um módulo definido com o nome de *run* e sua execução tem que estar contida nesse módulo. A classe ao ser chamada se inicializa, mas não executa. Para iniciar sua execução o método *run* deve ser invocado.

Na Figura 9, tem-se um exemplo de uma classe *Balao* que, ao ser inicializada, linha 20, o método *__init__* é executado recebendo atributos como: cor, tamanho e estourar. A partir da linha 21 do código, o método *run* é chamado e começa sua execução. O balão deve encher-se até que seu tamanho seja maior que 10, estourando ao atingir um tamanho superior. Um ponto importante é que o método *run* deve ter, em algum momento, uma condição de parada, caso contrário, será executado infinitamente. A Figura 10 mostra o resultado do processamento.

```
1 import threading, time
2
3 class Balao(threading.Thread):
4     def __init__(self):
5         self.cor = 'vermelho'
6         self.tamanho = 0
7         self.estourar = False
8         threading.Thread.__init__(self)
9     def encher(self, ar):
10        self.tamanho+=ar
11    def run(self):
12        while(not self.estourar):
13            print self.tamanho
14            self.encher(1)
15            time.sleep(1)
16            if self.tamanho > 10:
17                self.estourar = True
18
19
20 balao = Balao()
21 balao.start()
```

Figura 9 – Exemplo do uso do módulo *threading*

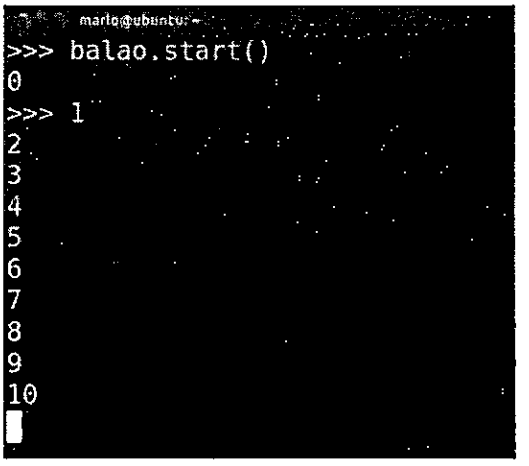

Figura 10 – Resultado da classe Balao após o uso do método *run*

2.4.5 Módulo *sys*

Módulo *sys* proporciona acesso a funções que interagem fortemente com o interpretador e com as variáveis usadas ou mantidas pelo mesmo. Manipulação dos argumentos da linha de comando como *argv[0]* e controle de entrada/saída de dados padrões do interpretador. A função usada foi *exit* (Figura 11), esta função tem como objetivo, terminar os processos do módulo do Python que a chamou, desse modo, matando todo e qualquer processo ou tarefa criados pelo interpretador.

Figura 11 – Módulo *sys*, função *exit*

2.5 MÓDULO *PYGAME*

Neste trabalho são necessários o reconhecimento de teclas do teclado, mouse ou joystick pressionadas por um usuário, captura de frames de uma webcam, suporte para criação de janelas que possam exibir imagens capturadas da câmera, transformar estas imagens capturadas em texto para que sejam enviadas pela rede, algumas funcionalidades básicas para a manipulação de imagens como carregar e salvar, além da possibilidade ser usado reprodução de arquivos de áudio. Bibliotecas voltadas para jogos englobam todas

essas funções e muito mais.

O módulo Pygame é responsável por estas tarefas à medida que, um jogo é executado em uma janela, ele reconhece eventos como botões pressionados, cliques do mouse, e ainda dá suporte para imagens e áudio.

Quadro 2 - Lista de funções definidas no módulo Pygame

Funções definidas dentro do módulo Pygame	Descrição
Init	Inicializa todos os módulos do Pygame importados
Locals	Contém várias constantes usadas pelo Pygame
Display	Controla a janela de display e a tela
Display.set_mode((640, 480))	Esta função cria uma *Surface* (superfície) de display com resolução de 640x480 pixels
Display.set_caption('PC - Cliente')	Altera o nome da janela do Pygame
Display.flip	Atualiza o conteúdo da janela
Image	Módulo que contém funções para carregar e salvar imagens. A imagem é carregada como objeto *Surface* e permite manipulações como desenhar linhas, captura de regiões, dentre outros
Image.load('camera_s.png')	Carrega uma imagem (PNG, GIF, BMP, PCX, TGA, TIF, LBM, PBM, XPM)
Image.save('teste.JPEG')	Salva uma imagem (BMP, TGA, PNG, JPEG)
Image.tostring(Surface, 'RGB')	Transforma uma imagem em texto
Image.fromstring(Surface, (320, 240), 'RGB')	Transforma texto da função *tostring* em imagem
Pygame.mixer	Módulo para carregar e reproduzir áudio
Mixer.music.load('capt_img.mp3')	Carrega um arquivo de áudio
Mixer.music.play	Reproduz um arquivo de áudio
Event	Módulo para interagir com eventos de dispositivos de entrada e filas
Event.get	Captura os eventos em uma fila
Key	Permite trabalhar com o teclado
Mouse.get_pos	Captura as coordenadas do mouse
Joystick.init	Inicializa o Joystick
Joystick.get_count	Retorna uma lista de endereço de joystick
Joystick.Joystick(0).init()	Inicializa o Joystick que está no primeiro índice da lista
Camera	Módulo para o uso da câmera
Time	Monitora o tempo
Quit	Finaliza todos os módulos do Pygame

Pygame é construída a partir de outra biblioteca de criação de jogos chamada *Simple Direct Media Layer* (SDL) comenta McGugan (2007). SDL foi escrita em C, uma linguagem geralmente usada em jogos por causa de sua velocidade e capacidade de se trabalhar com hardware em um nível baixo. Mas o desenvolvimento em C, ou seu sucessor C++, pode ser lento e sujeito a erro, McGugan (2007). Por ter seu desenvolvimento

baseado na SDL escrita em C, o Pygame tem um desempenho diferenciado a baixo nível, e acaba que, por se tornar ótima por trabalhar com dispositivos de entrada, mas pode tornar-se lenta ou propícia a erros.

A Figura 12 mostra o trecho final de uma imagem convertida em texto. Essa imagem é composta por um cabeçalho onde possui informações relacionadas a imagem como largura (*width*), altura (*height*), padrão de cores, entre outros, seguidos por uma sequência de caracteres que substituem os pixels.

No padrão de cores usado nesse trabalho, cada pixel é composto por três campos que representa as cores: vermelho, verde e azul (RGB) dependentes do padrão de cores usado na configuração da webcam. Cada cor recebe um valor que está compreendido de 0 a 255 onde, ao serem interpretados pela linguagem de programação, cada valor representará a intensidade de sua cor respectiva, compondo imagens de 24 bits de cores.

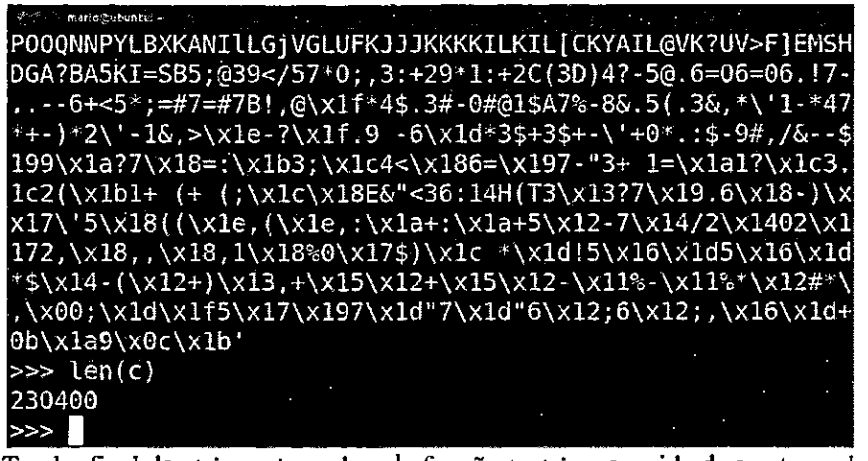

Figura 12 - Trecho final da *string* retornada pela função *tostring*, seguida de seu tamanho mostrado pela função *len*

2.6 OPENCV

Atualmente, a visão computacional, "conjunto de métodos e técnicas através dos quais sistemas computacionais podem ser capazes de interpretar imagens" (WANGENHEIM, 2007) impressiona a sociedade contemporânea. Dispondo de aplicações disponíveis no mercado que usam câmeras e inteligência artificial, a visão computacional identifica movimentos ou qualquer tipo de padrão. Um software que lida com imagens e vídeos e permite o monitoramento de áreas onde o usuário não está presente tem como função básica a renderização destas imagens criando, assim, um vídeo, uma vez que, o vídeo é composto de uma sequência destas imagens. A *Computer Vision Open Source*

(Opencv) é uma biblioteca multiplataforma. É escrita em C/C++ e permite sua integração com Python.

Segundo Bradski e Kaehler (2008, p.1), a biblioteca *Opencv*, contém mais de 500 funções, como exemplificado na Figura 13, que abrangem várias áreas na visão computacional, incluindo inspeção de fábricas de produtos, análises médicas, segurança, interface com o usuário, calibração da câmera, visão estéreo e robótica.

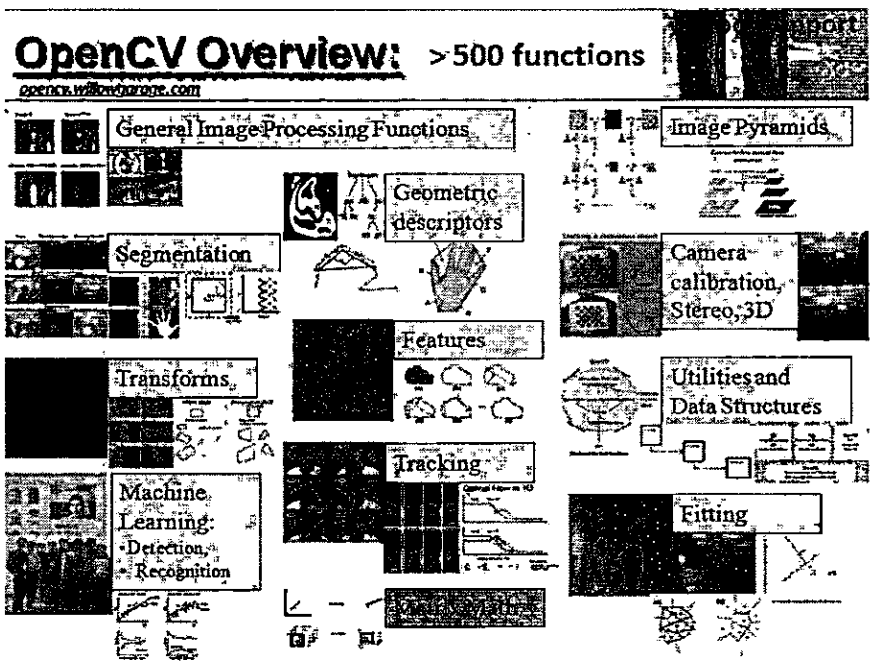

Figura 13 - Opencv
Fonte: opencv.willowgarage.com

Opencv possui a vantagem de permitir a interação com o Sistema Operacional, arquivos do sistema e hardware como câmeras com a biblioteca *highgui* (*high level graphical user interface*). Além de muitas outras funções, também permite a criação de um vídeo a partir de uma lista de imagens. Para isso são necessários dois passos básicos: a criação de uma tupla que contenha alguns dados para a renderização do vídeo e a função responsável por descarregar as imagens da memória para o disco rígido em forma de vídeo.

No manuseio de vídeos, sempre se leva em consideração o codec[3] para sua execução, o Opencv não é diferente. A função responsável por escrever um frame no arquivo recebe uma tupla com algumas informações sobre o vídeo, além da imagem a ser escrita.

[3] Codec é um software que é usado para compactar ou descompactar um arquivo de mídia digital, como música ou vídeo

O Opencv dispõe de uma função responsável pela renderização de imagens como vídeos, chamada Highgui.cvCreateVideoWriter composta por cinco índices: Nome do arquivo junto com a extensão, o codec do vídeo, outra tupla que especifica a resolução do vídeo, e um parâmetro booleano que faz com que o vídeo seja criado em cores ou escala de cinza.

- highgui.cvCreateVideoWriter('Nome.AVI', codec, fps, (resolução), True)
- 'Nome.AVI': String que identificará o arquivo criado junto com a extensão do arquivo;
- codec: Constante highgui.CV_FOURCC que especifica o codec de compressão do vídeo, recebe uma sequência de quatro caracteres que identificam o tipo de codec a ser usado. Os tipos são:
  - CV_FOURCC('P','I','M','1') = MPEG-1
  - CV_FOURCC('M','J','P','G') = motion-jpeg (não funciona bem)
  - CV_FOURCC('M','P','4','2') = MPEG-4.2
  - CV_FOURCC('D','I','V','3') = MPEG-4.3
  - CV_FOURCC('D','I','V','X') = MPEG-4
  - CV_FOURCC('U','2','6','3') = H263
  - CV_FOURCC('I','2','6','3') = H263I
  - CV_FOURCC('F','L','V','1') = FLV1
- fps: Quantidade de frames por segundo;
- resolução: Tupla contendo a resolução do vídeo, (320, 240);
- True: Se verdadeiro, o vídeo será criado em cores, caso contrário, será feito em escala de cinza.

O Opencv apresenta ainda a função Adaptors que permite a conversão explicita de Opencv imagens/matrizes para PIL e possui uma função chamada PIL2Ipl que converte uma imagem no formato PIL para o formato Opencv/IplcvMat, dando ao Opencv, no momento em que o usuário solicita a gravação de vídeo, o poder de ter uma imagem em formato não compatível e descarregá-la como arquivo de vídeo.

2.7 PIL

Python Imaging Library (PIL) atribui ao interpretador do Python a capacidade de realizar poderosos processamentos com imagens. Ideal para processamentos em quantidade e dá suporte na criação de miniaturas, conversão entre formatos de imagens,

impressão de imagens, etc.

> O PIL é um pacote de extensão de código aberto que adiciona ferramentas de processamento de imagens ao Python. Entre outras coisas, ele fornece ferramentas para miniaturas de imagens (*thumbnails*), transformar, conversões e estende o objeto básico imagem do tkinter, adicionando suporte a exibição de muitos tipos de arquivos de imagem. PIL permite exibir imagens de formatos JPEG, TIFF, e PNG que não são suportados pelo kit de ferramentas base do tkinter em si (sem extensão, tkinter suporta GIFs e alguns de formatos bitmap). (LUTZ, 2011, p.366, traduzido pelo autor)

PIL tem uma classe chamada *Image*, uma das classes mais importantes e definida em um módulo com o mesmo nome. Com ela pode-se criar instâncias dessa classe de várias maneiras, carregando arquivos de imagem, com resultado de processamento de outras imagens, etc. Além disso, *Image* é essencial no momento em que se tem uma imagem capturada pelo Pygame, e transformar essa imagem em uma classe aceita pelo Opencv.

Ao receber uma imagem pela rede, o computador Cliente obtém uma imagem no formato *Image Surface*, por não ser originária do Python, Opencv não reconhece esse tipo de imagem, a classe *Image* se responsabiliza por montar a imagem no formato PIL reconhecível pelo Opencv, possibilitando sua conversão de tipos e, posteriormente, a gravação do vídeo.

2.8 MÓDULO TKINTER

Tomando como referência um computador atual, com efeitos visuais bem elaborados, com animações que melhoram o visual dos programas, com displays e placas de vídeo que dão suporte a 32 bits de cores, fica às vezes não perceptível a gama de conhecimentos necessários para alcançar essa tecnologia visual, que vem com o propósito de diminuir a dificuldade de interação com a máquina.

Poucos usuários de computador têm dificuldades de interagir com os programas, afirma Budd (2009), isso se deve a Interface com o Usuário (IU), no desenvolvimento de um programa, principalmente destinados a usuários domésticos, itens como janelas, menus, botões, caixas de texto, barras de rolagem, etc., são indispensáveis para um programa. O Python não dispõe de módulos em sua biblioteca padrão para criar componentes de interface, mas existem várias bibliotecas diferentes que podem ser

adaptadas para o Python.

Hoje em dia, a IU é parte essencial de um programa, principalmente voltado a usuário doméstico. O Python tem o Tkinter como módulo responsável pela comunicação humano-computador, com objetivo de tornar essa relação o mais amigável possível. O Tkinter permite ao programador usar *widgets*[4] como *Frame*, *Label*, *Entry* (caixa de entrada de texto), *Buttons* (botões), *Text* (áreas de texto), *Canvas*, *Radiobutton*, *Checkbutton*, *Scale*, *Listbox*, *Scrollbox*, *OptionMenu*, *Spinbox*, *LabelFrame* e *PanedWindow*, além de fornecer classes que dão suporte a exibição, posicionamento e controle dessas *widgets*. Isso no intuito de propor uma melhor interação com o usuário.

Em se tratando de IU, multimídias visuais são indispensáveis. Neste sentido, o Tkinter conta com um complemento interessante. ImageTk é um exemplo disso, já que consiste num módulo que permite o uso de imagens no formato PIL em janelas originadas pelo Tkinter, a fim de propor um layout mais atrativo e amigável para o usuário.

2.8.1 Tkinter, classes e funções

O Quadro 3, na página 34, mostra uma lista de classes e funções mais usadas no desenvolvimento do software proposto neste trabalho, definidas dentro do módulo Tkinter, acoplada a suas descrições.

2.9 MÓDULO SOCKET

Falar de rede é perceber que antes de sua existência houveram sucessivas transformações. Graças a inquietação do homem diante do mundo, foi possível despertar desde os primórdios curiosidades para o surgimento do conhecimento. Primeiro, na idade antiga, com o domínio do fogo, em seguida, na idade média, com estudo de estratégias para as guerras, mais tarde com a chegada do Renascimento, que origina, de certa forma, os primeiros reflexos para o surgimento da tecnologia. Tanenbaum (2002, p.18) expõe justamente esses reflexos históricos sobre os avanços tecnológicos ao afirmar que:

> O Século XVIII foi a época dos grandes sistemas mecânicos que acompanharam a Revolução Industrial. O Século XIX foi a era das máquinas a vapor. As principais conquistas tecnológicas do Século XX se

[4] Para mais detalhes a documentação da biblioteca deve ser consultada.

> deram no campo da aquisição, do processamento e da distribuição de informações. Entre outros desenvolvimentos, vimos a instalação das redes de telefonia em escala mundial, a invenção do rádio e da televisão, o nascimento e o crescimento sem precedentes da indústria de informática e o lançamento dos satélites de comunicação. Como resultado do rápido progresso tecnológico, essas áreas estão convergindo rapidamente e são cada vez menores as diferenças entre coleta, transporte, armazenamento e processamento de informações. Organizações com centenas de escritórios dispersos por uma extensa área geográfica podem, com um simples apertar de um botão, examinar o status atual de suas filiais mais remotas. A medida que cresce nossa capacidade de colher, processar e distribuir informações, torna-se ainda maior a demanda por formas de processamento de informações ainda mais sofisticadas

Quadro 3 - Lista de funções do Tkinter

Funções definidas dentro do módulo Tkinter	Descrição
TK()	Instância da classe Tkinter
Tk().title()	Atribui um título a janela do Tkinter
Tk().geometry()	Recebe uma *string* que representará a geometria da janela, ou seja, largura, altura, e posição na tela a ser iniciada
Tk().mainloop()	Função que deve ser chamada assim que a janela for completamente montada. Mantém a janela em um *loop* infinito até que seja chamada a função *destroy*
tkFileDialog	Proporciona interface com diálogos de arquivos nativos
tkFileDialog.askdirectory()	Abre uma janela de diálogo para a busca de um diretório e retorna um nome de arquivo
tkMessageBox	Exibe caixas de mensagens
tkMessageBox.showerror	Exibe uma caixa de mensagem de erro
Label	*Widget* que pode exibir texto na janela
Entry	Permite mostrar uma caixa de texto simples
Button	*Widget* botão
Widget.pack()	Atribui *widget's* à janela principal, separando-as por blocos
Widget.place()	Atribui *widget's* à janela principal em uma posição específica
Tk().destroy()	Encerra a aplicação Tkinter, destrói a janela atual e todos seus descendentes

Percebe-se diante do exposto que já faz parte da sociedade o comunicar-se ou transmitir informações de maneira sofisticada, prática e rápida. A rede, nesse sentido, veio a contribuir significativamente para a globalização de transferência de dados.

Etimologicamente a palavra, Rede é originária do latim *rete, is* que significa "rede ou teia", o que denota o entrelaçamento de fios, isso no sentido de origem da palavra. Entretanto, ao longo dos anos esse conceito ganhou novas roupagens, passando a ser reformulado nos dias atuais com um sentido mais abstrato e metafórico, ou seja,

significando relações de diversas naturezas, estas que englobam as áreas, não apenas das ciências exatas, mas também das ciências humanas. Para reafirmar o exposto Busch (2008) disserta que:

> A palavra rede é bem antiga e vem do latim retis, significando entrelaçamento de fios com aberturas regulares que formam uma espécie de tecido. A partir da noção de entrelaçamento, malha e estrutura reticulada, a palavra rede foi ganhando novos significados ao longo dos tempos, passando a ser empregada em diferentes situações.

Diante das reflexões expostas, percebe-se que o sentido de rede é um termo genérico que define tanto pessoas quanto objetos. Contudo, o presente trabalho visa abordar especificamente o sentido de rede de origem, isto é, aquele voltado para o entrelaçamento de fios, nós, malhas, tecidos, que unidos desempenham uma ação interativa entre *hosts,* ou seja, aquele voltado para o conjunto de computadores e periféricos conectados uns aos outros, na solução proposta, interagindo entre si, através de sockets.

> Python vem com módulos de Internet padrões que permitem programas em Python realizarem uma grande variedade de tarefas com redes em modo cliente e servidor. Os script podem se comunicar através dos socket; extrair informações dos formulário enviados para os scripts do servidor CGI; transferência de arquivos por FTP; gerar e analisar arquivos XML; enviar, receber, compor e analisar e-mail; buscar páginas web por URL; analisar o HTML e XML de páginas web buscadas; comunicação através de XML-RPC, SOAP[5], Telnet dentre outros. As bibliotecas do Python realizam essas tarefas de forma relativamente simples. (LUTZ, 2009, p.10, traduzido pelo autor).

Socket (biblioteca) é um módulo do Python que proporciona operações com *socket* (tipo de comunicação), dando assim, a possibilidade de se estabelecer comunicação entre dois ou mais computadores através de uma rede, dando suporte ao desenvolvimento de programas de arquitetura Cliente/Servidor, de maneira em quer computadores na rede possa oferecer serviços, desempenhando o papel de servidores, além de outros computadores da rede que fazem requisições/solicitações a esses serviços passando, assim, a serem chamados de clientes. Segundo Keel (2013): "Socket é um termo usado para identificar um ponto de ligação que qualquer programa de computador pode usar para transmitir dados através da Internet".

---

[5] Protocolo simples baseado em XML que permite as aplicações a troca de informações através de HTTP.

A configuração correta do socket é primordial para seu bom funcionamento, sendo se suma importância o uso correto do endereço IP e a porta de comunicação. Configurados corretamente, faz-se possível o a comunicação entre os computadores, sendo permitido o envio e recebimento de dados por ambos.

Quadro 4 - Funções e constantes definidas dentro do módulo socket

Funções e constantes definidas dentro do módulo socket	Descrição
socket()	Cria um novo objeto socket
socketpair()	Cria um par de novos objetos socket*
gethostname()	Retorna o nome do host atual
socket.setdefaulttimeout()	Atribui um valor de tempo limite para espera
create_connection()	Conecta a um endereço, com um tempo limite opcional a endereço fonte opcional
AF_INET, AF_UNIX	Domínios de socket (primeiro argumento na chamada da função socket())
SOCK_STREAM, SOCKDGRAM, SOCK_RAW	Tipos de socket (segundo argumento da função socket())
accept()	Aceita uma conexão retornando um novo socket e o endereço do cliente
bind(endereço)	Vincula o socket a um endereço local
close()	Fecha o socket
connect(endereço)	Conecta o socket a um endereço remoto
getpeername()	Retorna o endereço remoto*
getsockname()	retorna o endereço local
listen()	Começa a escutar tentativas de conexões
recv()	Recebe dados
send()	Envia dados
shutdown()	Encerra o tráfego de dados em ambas as direções

* Não disponível para todas as plataformas

As Figuras 14 e 15, a seguir, mostram um exemplo de código simples na programação de um socket servidor e outro socket que irá desempenhar o papel de cliente.

Ao analisar o código da Figura 14, pode ser observado, na linha 1, o modo de importação do módulo socket. Para se instanciar um objeto socket, chama-se a função *socket()*, na linha 3, dando-a dois argumentos. O primeiro é o domínio, representado pela constante AF_INET significando a família de endereços a ser usada no socket que, no caso, IPV4, e o segundo argumento é passado a constante SOCK_STREAM especificando que o protocolo que o socket usará será o TCP. No servidor um endereço de IP e uma porta de comunicação, onde o mesmo ouvirá os clientes. Para isso é usada a função *bind()* como exemplificado na linha 4, além disso, e exigido que seja configurada a quantidade de clientes que este servidor irá ouvir através da função *listen()*, linha 5.

```
1  import socket
2
3  servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  servidor.bind(('localhost', 8089))
5  servidor.listen(5)
6
7  while True:
8      conexao, endereco = servidor.accept()
9      dado = conexao.recv(64)
10     if len(dado) > 0:
11         print dado
12         break
```

Figura 14 - Exemplo de socket servidor

O servidor somente aceita solicitações de conexão, quem enviará essas solicitações é o cliente. Na linha 8 é usada a função *accept()* usada para aceitar conexões e na linha 9 o uso da função *recv()* permite ao socket receber algum dado, sendo que, por sugestão da própria biblioteca socket, deve ser passado por parâmetro um número com base potencial 2, que representa a quantidade, em bytes, a ser capturada da rede.

Já na Figura 15 o cliente, até o momento em que chama-se a função *socket()*, é exatamente igual ao servidor. Sendo papel do cliente enviar a requisição de conexão, a linha 4 ilustra o exemplo da função *connect()*, que pede uma tupla, como parâmetro, com dois índices. O primeiro deve ser um endereço IP, mais especificamente o endereço de IP do servidor, seguido de um número que representará a porta de comunicação que deve ser a mesma porta configurada no servidor. Na linha 5, o uso da função *send()* mostra como enviar dados com o socket, sendo que, tanto o servidor quanto o cliente, após o sucesso da tentativa de conexão, podem fazer uso das funções *send* e *recv*.

```
1  import socket
2
3  cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  cliente.connect(('localhost', 8089))
5  cliente.send('hello')
```

Figura 15 - Exemplo de socket cliente

Para encerrar a conexão, faz-se preciso o uso da função *shutdown()*, onde qualquer um dos sockets, cliente ou servidor, podem invocá-la. Caso a função seja chamada o tráfego de dados será encerrado de ambos os lados.

2.10 ARDUINO

A massa de informações circundantes e a necessidade tão emergencial do ser humano em descobrir todos os mistérios e porquês do mundo têm sido elementos indispensáveis na construção do saber científico, muito em voga nos dias atuais. A ciência pautada nas novas experiências e ancorada na tecnologia tem proporcionado cenas inimagináveis aos seres, como viagens espaciais, avanços científicos nos mais diversos campos e em especial as que viabilizam uma melhor condição de vida ao ser. Conhecimentos e materiais, outrora tão rudimentares e inacessíveis, hoje exploram a praticidade e agregam usuários, estudiosos e fãs. Neste mesmo sentido os conhecimentos construídos acerca da tecnologia e seu uso desenfreado proporcionam novos saberes e agregam a cognição humana novas concepções de ser e agir. O computador de certa forma se assemelha ao homem, uma vez que possui o aspecto cognitivo. Cavalcante (2011) aponta que:

> O computador é uma importante ferramenta cognitiva, isto é, permite ao estudante desenvolver habilidades, interiorizar conhecimentos e organizá-los de modo a construir uma interpretação do mundo que o cerca.

A proposta apresentada neste trabalho, além do software, consiste na montagem de um equipamento robótico (um protótipo) elaborado a partir de um chassi de acrílico com motores acoplados aos pneus e placas de circuitos responsáveis por controlar estes motores e, consequentemente, proporcionar liberdade de movimentação à estrutura robótica. Visando reduzir custos e/ou outras dificuldades na montagem do esqueleto do equipamento, deu-se uma maior importância a hardwares livres. Dentre as existentes, como o PIC, por exemplo, foi escolhido o Arduino, pois para utilizar o PIC é necessário fazer uma placa de circuito para usá-lo, ou seja, é preciso montar o equipamento enquanto o Arduino já vem pronto para ser programado e utilizado. Além disso, como afirma Cavalcante (2011):

> A placa Arduino vem sendo utilizada com muito sucesso. A plataforma para o desenvolvimento dos programas de controle está disponível na Internet e existem diferentes versões do circuito no mercado nacional por preços acessíveis quando comparados às interfaces disponibilizadas no mercado por empresas como CIDEPE, Pasco Scientific e Phywe, por exemplo.

O Arduino, exibido na Figura 16, é uma ferramenta capaz de controlar, executar e interagir com o mundo físico, através de sensores e atuadores. O grande diferencial desta ferramenta é a sua utilidade e praticidade, pois são acessíveis e apresentam baixo custo, bem como seu manuseio é, de certa maneira, de fácil acesso, tanto para experientes na área, quanto para amadores do ramo computacional.

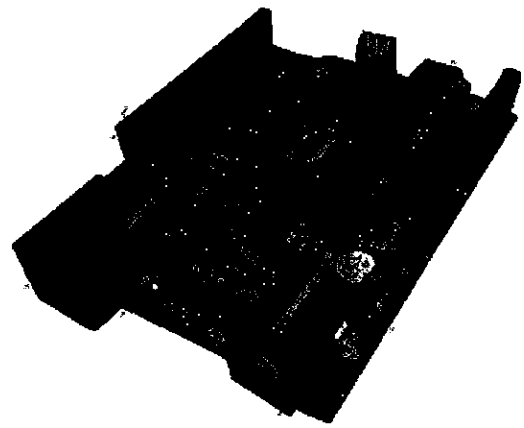

Figura 16 - Arduino

Representando uma Plataforma física de computação livre, o Arduino é baseado numa placa de circuitos com um micro controlador, que pode ser programado na linguagem C de acordo com a função que executará, permitindo assim, a interação do software com o ambiente, através de sensores e atuadores.

Dentre as aplicabilidades do Arduino é possível citar o desenvolvimento de objetos interativos, estes que admitem:

> Acoplagem de uma série de sensores ou chaves, que controlam uma variedade de luzes, motores ou outras saídas físicas. Projetos do Arduino podem ser independentes, ou podem se comunicar com software rodando em seu computador (como Flash, Processing, MaxMSP.). Os circuitos podem ser montados à mão ou comprados pré-montados; o software de programação de código-livre pode ser baixado de graça. (ARAÚJO, 2009).

Diante do exposto, observa-se que o Arduino veio a contribuir para o avanço tecnológico nos diversos sentidos, na medida em que propicia baixo custo, fácil acesso, além de ser uma plataforma que pode executar ações, de forma precisa e objetiva, unido a controladores de um ambiente, principalmente hostil, sendo muito útil em prototipagem de projetos.

Convém mencionar que o Arduino pode ser projetado e adaptado às necessidades de seu programador, por meio do incremento de placas chamadas *shields*, que são responsáveis pelo exercício de tarefas específicas como comunicação através de rede, seja ela com fio ou não, sensores de umidade, temperatura, rotação, aceleração, distância, manipulação de voltagens grandes o suficiente para danificá-lo usando relés, controladores de motores de passo ou de corrente contínua, dentre outros.

O Arduino é utilizado para realizar tarefas de automação e robótica, possui um chip e que funciona como um computador completo com menor poder de processamento e memória, facilmente programável em C e através de *shields*, que são placas menores, que podem ser integradas a ele para executar tarefas, podendo-se obter uma variedade de capacidades, desde uma simples ação de ligar uma pequena lâmpada, até automatizar uma casa inteira, saber sua localização global, entre outros.

3 MODELAGEM

Por ter a possibilidade de um programa ter seu nível de complexidade elevado, uma vez que é necessário a conexão entre hardware e software, pode ser difícil o controle dos componentes utilizados por seus desenvolvedores. Neste sentido, a UML apresenta-se como uma linguagem de modelagem de softwares e desenvolvimento de sistemas, visando gerenciar, de uma melhor maneira, toda essa complexidade e, por fim, proporcionar um melhor entendimento, em uma visão geral, do sistema ou software a ser desenvolvido. Para Martinez (2010):

> A UML (Unified Modeling Language), que significa Linguagem Unificada de Modelagem é uma linguagem padrão para modelagem orientada a objetos. Ela surgiu da fusão de três grandes métodos, do BOOCH, OMT (Rumbaugh) e OOSE (Jacobson). Esta linguagem de modelagem não proprietária de terceira geração, não é um método de desenvolvimento. Têm como papel auxiliar a visualizar o desenho e a comunicação entre objetos. Ela permite que desenvolvedores visualizem os produtos de seu trabalho em diagramas padronizados, e é muito usada para criar modelos de sistemas de software.

Para isso, a UML disponibiliza vários diagramas que devem descrever o programa que será desenvolvido, sendo possível que até mesmo um leigo na área da informática possa compreender o sistema. O Quadro5 mostra a lista de alguns dos diagramas da UML.

Quadro 5 - Tipos de Diagramas da UML

Tipo de Diagrama	Descrição
Diagrama de Caso de Uso	Mostra o possível comportamento do sistema, interações entre o software e os usuários ou outros sistemas. Muito útil na hora do mapeamento dos requisitos do sistema
Diagrama de Classe	Define a estrutura de classes, tipos, interfaces do sistema e seus relacionamentos
Diagrama de Sequencia	Ilustra interações entre objetos preocupando-se com a ordem temporal da troca de mensagens
Diagrama de Atividades	Descreve os passos percorridos para a conclusão de uma atividade sequencial ou paralela dentro do sistema

Este capítulo mostra uma melhor visão da proposta apresentada neste trabalho

exibindo o diagrama de caso de uso, diagrama de classe e o diagrama de atividades, para melhor entendimento do funcionamento do sistema.

Este software e composto por dois módulos:

- Cliente
  - Conecta-se ao servidor
  - Salva uma imagem capturada
  - Salva vídeos
  - Envia comandos ao servidor como resposta
- Servidor
  - Permite a conexão de clientes
  - Envia imagens ao cliente
  - Recebe comandos respostas do cliente

3.1 DIAGRAMAS DE CASOS DE USO

O Diagrama de Caso de Uso, que geralmente é feito no início da modelagem, tenta possibilitar a compreensão do sistema por qualquer pessoa por meio de uma linguagem simples, sendo flexível e informal, mostrando os serviços e operações prestado a cada usuário sem preocupar-se com funções que deverão ser desenvolvidas. Além disso, é composto por dois itens principais: Atores e Casos de Uso.

3.1.1 Atores

Os Atores representam usuários que terão, de alguma forma, que interagir com o sistema, equipamentos de hardware ou até mesmo outro sistema, ou seja, um item que não faz parte do sistema, mas, quem ou o que vai interagir com o programa. É representado graficamente por "bonecos magros" com uma descrição simples logo abaixo. A Figura 17 ilustra um exemplo de Ator no Diagrama de Casos de Uso.

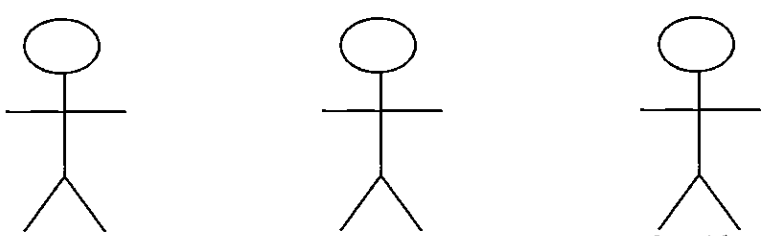

Figura 17 - Exemplo de Atores no Diagrama de Casos de Uso

3.1.2 Casos de Uso

Os Casos de Uso, no diagrama, ilustra o que o sistema deve fazer como: serviços a serem prestados, tarefas e/ou funções a serem executadas. A representação gráfica do Caso de Uso é uma elipse que contem em seu interior, uma pequena descrição do serviço que o mesmo se refere. A Figura 18 trás um exemplo de Caso de Uso. Cabe ressaltar que a descrição/nome do caso de uso seja um verbo no infinitivo como: transmitir, pagar, comprar, entre outros.

Figura 18 - Exemplo de Caso de Uso

3.1.3 Relações entre os Atores os Casos de Uso

Além dos Atores e dos Casos de Uso, faz-se necessário ilustrar no diagrama a relação entre eles demonstrando que os Atores utilizam as funções que o Caso de Uso representa. No Diagrama de Casos de Uso algumas interações ou relacionamentos entre seus Atores ou Casos de Usos são chamadas de Associações. As associações podem ser de inclusão ou extensão.

A associação de Inclusão é usada quando dois ou mais Casos de Uso apresentam relação de obrigatoriedade, ou seja, a execução do primeiro implica/obriga que o segundo seja executado também. É representado por uma seta tracejada nomeada por *include*, esta que deve ser escrita com dois sinais de menor que (<) antes e dois sinais de maior que (>) depois como mostra a Figura 19.

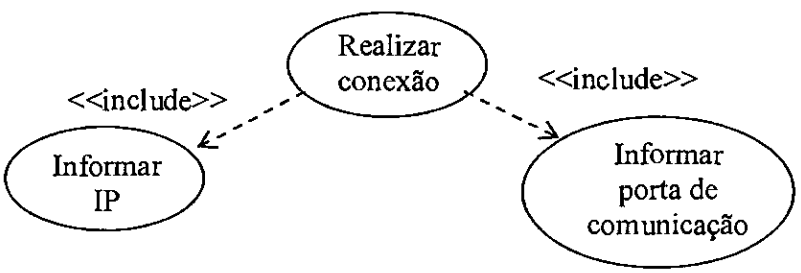

Figura 19 - Exemplo de Inclusão

A associação de Extensão indica que a relação somente ocorrerá se uma condição for satisfeita ou ocorra uma situação específica. Sua representação gráfica é

semelhante a Inclusão, mas com a palavra *extend* ao lado da seta ilustrado na Figura 20.

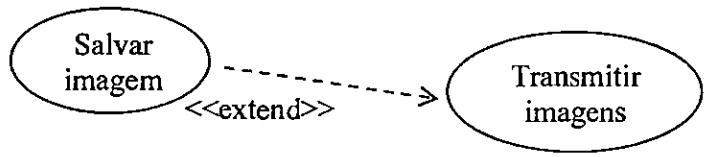

Figura 20 - Exemplo de Extensão

3.2 DIAGRAMAS DE CASOS DE USO PROPOSTO

Para que o software realize sua tarefa, e necessário que o usuário informe um endereço de IP e um número de porta de comunicação. Assim que a conexão é estabelecida, a versão CLIENTE, deverá permitir o usuário a:

- Salvar uma imagem recebida do servidor
- Gravar um vídeo a partir das imagens recebidas
- Escolher o local onde o software deve salvar as imagens
- Escolher o local onde o software deve salvar os vídeos

A Figura 21 ilustra o modelo de caso de uso do sistema, onde o objetivo mantém-se em torno da transmissão de imagens, sendo obrigatório se estabelecer/realizar a conexão entre as versões Cliente e Servidor, além disso, e necessário que ambos sejam configurados com o mesmo endereço de IP, mais especificamente o IP do servidor, e porta de comunicação.

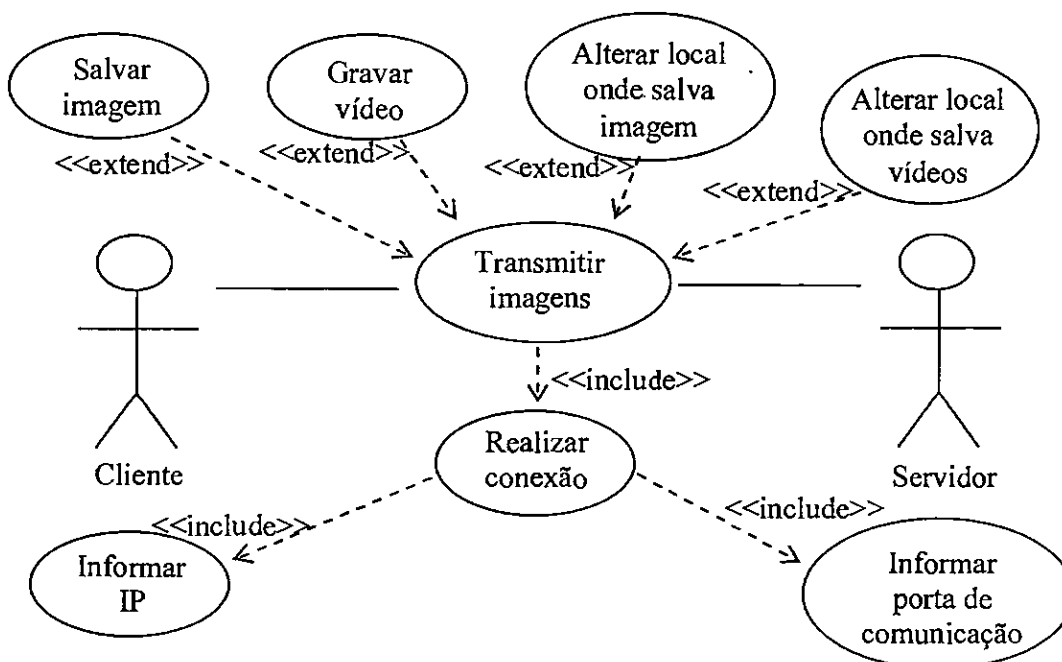

Figura 21 - Diagrama de Caso de Uso

3.3 DIAGRAMA DE CLASSES

O Diagrama de Classe é considerado o mais importante diagrama da UML, permite a visualização mais detalhada da estrutura funcional do sistema desenvolvido ilustrando suas classes, objetos, funções e as relações entre os mesmos.

As classes são representadas por retângulos divididos em três partes onde a primeira deve conter o nome da classe, a intermediária os atributos dessa classe e a inferior contendo os métodos da mesma. O sinal existente antes de cada nome de atributo ou método indica suas visibilidades, ou seja, se eles são Públicos (+), Protegidos (#), Privados (-) e em Pacotes (~). Não diferente do Diagrama de Casos de Uso, também se faz necessário ilustrar a relação entre essas classes.

As associações necessárias no Diagrama de Classe do software proposto são a Dependência, por ter classes que dependem de outras, representada por uma reta tracejada contento uma seta na extremidade da classe dependente, e a Agregação significando que uma classe, objeto-parte, precisa ser complementada por informações de uma outra classe chamada objeto-todo, onde seu símbolo e composto por uma reta com um losango na extremidade do objeto-todo ilustrados na Figura 22.

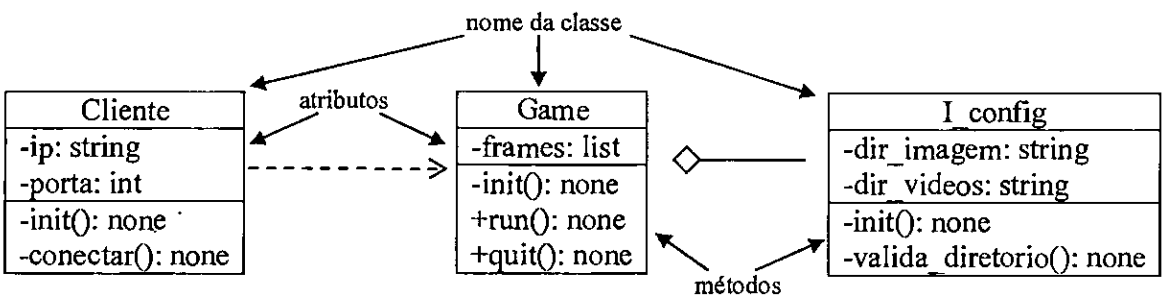

Figura 22 - Exemplo de Diagrama de Classe

3.4 DIAGRAMA DE CLASSE PROJETO CONTROLE

No software proposto neste trabalho, identifica-se a necessidade do uso das seguintes classes: Servidor, Cliente, I_config, Cliente_config e Game explicadas na Figura 23 que têm como objetivo a ilustração do diagrama de classe deste programa. Pode-se observar que o Servidor é relativamente simples, desenvolvido em Python, bastando somente uma classe servidor. Já o Cliente exige um pouco mais de cuidado em sua análise.

A versão Cliente sugerida, possui inicialmente uma classe que inicialize a

janela de configuração do socket, Cliente_config, assim que o usuário solicitar que a conexão seja estabelecida, será instanciada a classe Cliente que contém o objeto socket. Caso a tentativa de conexão tenha sucesso mais uma classe é chamada, a classe Game que se responsabilizará por exibir na tela as imagens recebidas, além de possibilitar que o usuário salve imagens ou vídeos com o auxílio de uma classe adicional chamada I_config.

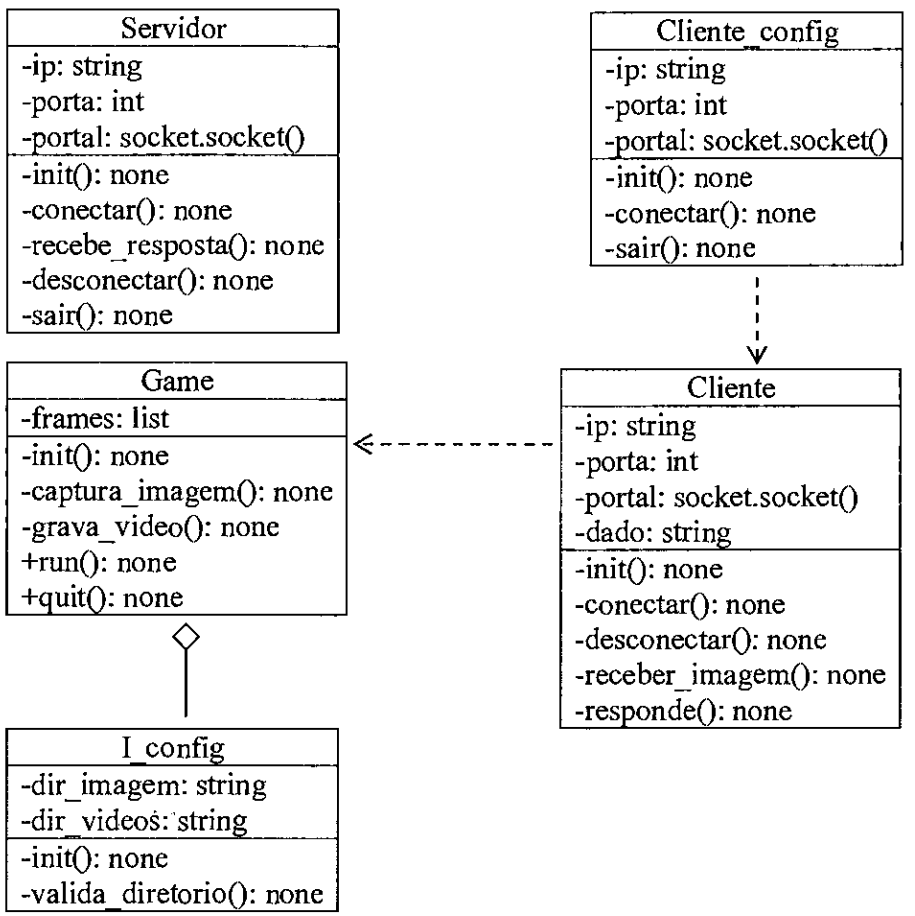

Figura 23 - Diagrama de Classes

3.5 DIAGRAMA DE ATIVIDADES

O Diagrama de Atividades é constituído pelas ações e fluxos lógico de execução do programa e vem com o objetivo de ilustrar esse comportamento. As atividades são etapas de processos a serem realizados sequencialmente interligados por uma seta chamadas de Transição, ilustrados na Figura 24, que pode vir atrelada a condições representadas graficamente por uma frase entre colchetes "[]", descrevendo essas condições próximas as Transições respectivas.

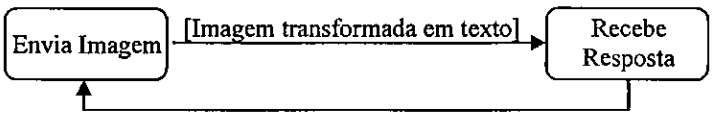

Figura 24 - Exemplo de Atividade

3.5.1 Características do Diagrama de Atividades

O Diagrama de Atividades define um ponto de início e de termino do processo, além disso, há representações gráficas da situação em que o estado atual possui mais de uma possibilidade de continuação do processo. Isso é feito com o ponto Merge constituído por um losango. A Concorrência de processos também pode ser ilustrada no Diagrama de Atividades com bifurcações de Transições, onde existirá a possibilidade de uma Transição gerar outras Transições, sendo assim, os processos representados por essas Transições serão executados de forma concorrente pelo computador. A Figura 25 mostra alguns dos principais pontos do Diagrama de Atividades.

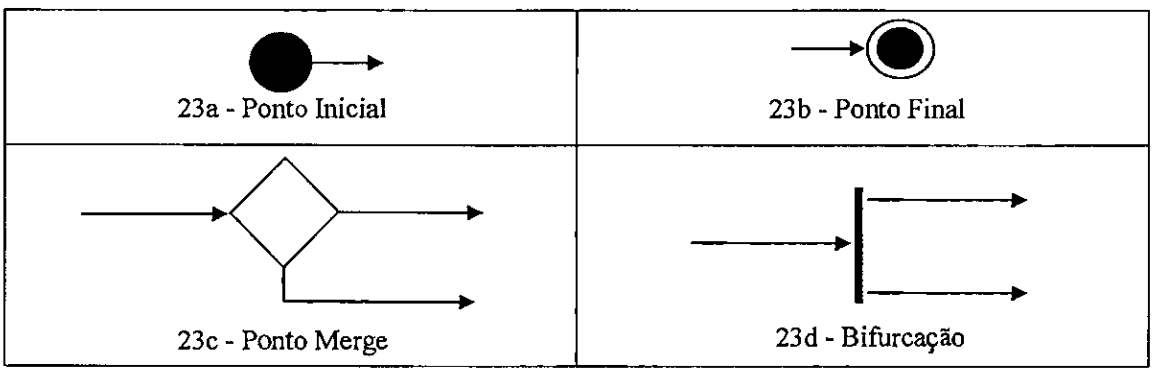

Figura 25 - Situações do Diagrama de Atividades

3.6 DIAGRAMA DE ATIVIDADES DO SERVIDOR

A Figura 26 ilustra o Diagrama de Atividades do programa Servidor. A partir de seu ponto inicial, faz-se necessário que sejam informado o endereço IP do próprio Servidor e a porta de comunicação que o mesmo escutará. Após a configuração, o Servidor entra em modo de escuta à espera da solicitação de conexão do Cliente.

Assim que o Cliente solicita conexão, e esta for estabelecida, o Servidor captura uma imagem da câmera e a transmite para o Cliente. Depois que se transmite a imagem, o Servidor volta ao estado de escuta, esperando uma resposta do Cliente,

entretanto, pode haver duas opções de respostas: a solicitação de uma nova imagem ou pedido de desconexão, nesse caso, o ponto final de execução do servidor.

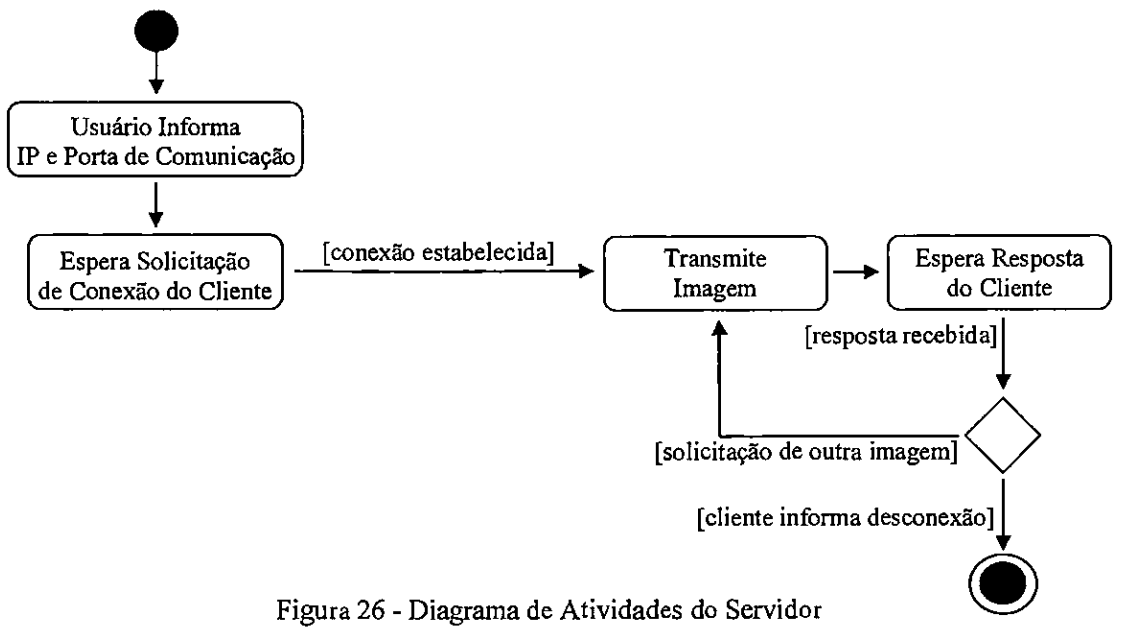

Figura 26 - Diagrama de Atividades do Servidor

3.7 DIAGRAMA DE ATIVIDADES DO CLIENTE

O Diagrama de Atividades do Cliente possui algumas diferenças em alguns pontos (Figura 27).

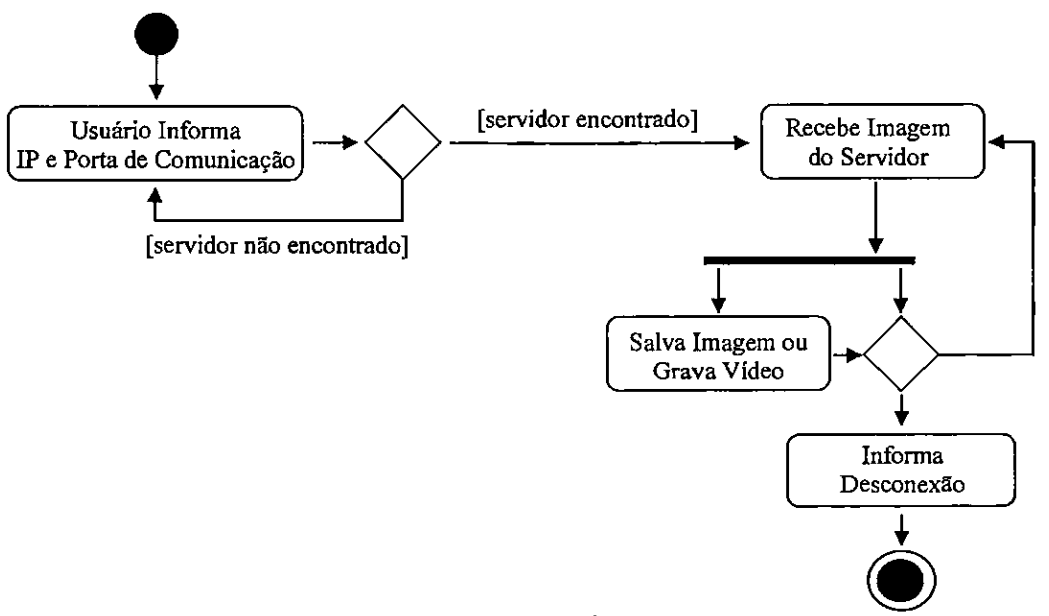

Figura 27 - Diagrama de Atividades Cliente

Assim que sai do ponto inicial, o Cliente solicita os mesmos dados de configuração do Servidor: endereço de IP do servidor e Porta de Comunicação. Em seguida, o Cliente tenta a conexão com o Servidor, caso não consiga, serão solicitados IP e a Porta novamente. Se a conexão obtiver sucesso, o Cliente se prepara para receber a imagem do Servidor e a exibe para o usuário.

Quando o programa está em execução, é necessário que seja possível que o usuário possa salvar o frame atual como imagem ou capturar vários quadros para renderizá-los como vídeo.

O programa Cliente só atinge seu ponto final assim que solicitado pelo usuário. Caso isso ocorra, será enviado o comando de desconexão ao servidor, desse modo, encerrando a conexão.

4 ESTRUTURA E ARQUITETURA

O projeto desenvolvido neste trabalho, intitulado, Projeto Controle, consiste num sistema criado para permitir a interação de um usuário controlador com um ambiente qualquer remotamente. De maneira que o usuário deve estar em posse de um computador conectado com um cabo de ethernet a um roteador sem fio. Que por sua vez deve estar conectado a um outro computador através da rede wireless, este, com uma câmera plugada para se extrair as imagens a serem enviadas. Assim ilustra a Figura 28.

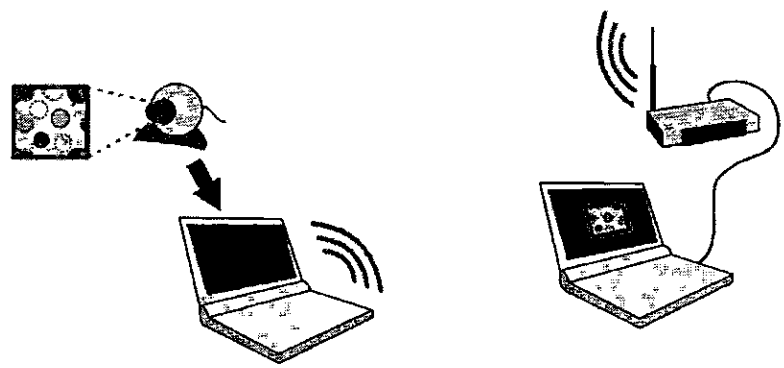

Figura 28 - Arquitetura física do Projeto Controle

No desenvolvimento do trabalho, de início foi necessário pesquisar sobre duas ações diferentes, que são básicas para o funcionamento do projeto com sucesso. As ações capturaram imagens da webcam e realizaram comunicação entre dois computadores através da rede, com o uso do Python. Estas ações foram escolhidas para este estudo, por curiosidade e pelo desafio de se aprofundar na linguagem de programação Python.

Ao longo do experimento, surgiu a ideia de integrar os estudos de captura de imagens ao de comunicação entre computadores, ou seja, desenvolveu-se toda a parte lógica da pesquisa que ao ser concluída, pode ser aplicada em diversas situações no uso de monitoramento com câmeras. É possível o uso da aplicação desenvolvida em sistemas de segurança com visualização em tempo real através da internet como sistema de trânsito ou sistemas internos de TV. Pode ser empregado, também, no desarmamento de bombas, análise de construções com risco de desabamento, cirurgias realizadas de maneira remota (Telecirurgia), dentre outros.

A aplicação desenvolvida necessita de uma configuração para ser executada. A mesma exige que seja informado um endereço de IP e uma porta de comunicação para ser estabelecida a conexão entre Cliente (Figura 29) e Servidor (Figura 30). Contudo, a versão

Servidor disponibiliza, a quem o configura, informações que identificam se a webcam e o Arduino estão conectados e configurados corretamente.

O Projeto Controle Cliente possui dois campos onde o usuário deve digitar o IP que foi colocado no computador hospedeiro da versão Servidor e a porta de comunicação respectivamente. Após informar os dados anteriores, o botão OK deve ser clicado para realizar a tentativa de conexão. Se a tentativa de conexão obtiver sucesso, essa janela inicial será desativada e será aberta uma nova janela (Figura 31 na página 53), melhor explicada mais adiante, onde serão exibidos as imagens recebidas pelo Servidor e alguns botões para salvar imagem e outros para gravar vídeo e interromper essa gravação. Caso for necessário encerrar o programa, o botão Sair deverá ser acionado.

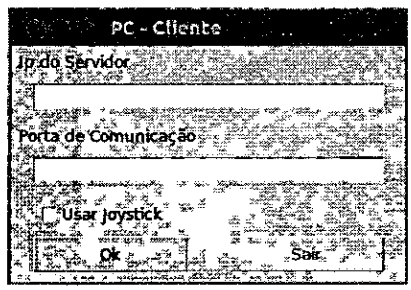

Figura 29 - Projeto Controle Cliente

A versão Servidor do programa também possui campos para que sejam informados o IP e a porta de comunicação. Ainda assim, possui um botão Detectar Dispositivo que, ao ser clicado, a webcam é configurada automaticamente e assim mostrado o caminho do dispositivo de vídeo no campo Dispositivo de Captura de Vídeo. O botão Detectar Arduino ativa uma função que configura o Arduino onde o símbolo dele se mantém monocromático enquanto o mesmo não estiver pronto para executar. Este símbolo fica da cor azul assim que o software julga que o Arduino está pronto para funcionar.

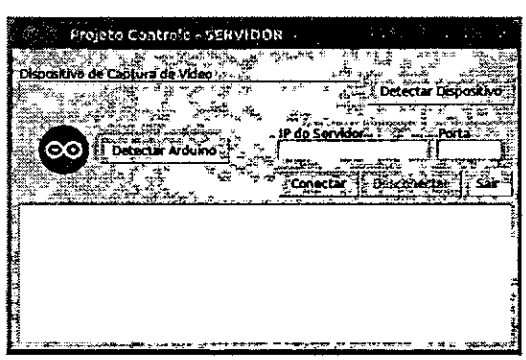

Figura 30 - Projeto Controle Servidor

A tela de execução do Projeto Controle Cliente (Figura 31) consiste em uma janela onde serão exibidas as imagens recebidas. A mesma possui, ainda, cinco botões onde suas funções são: salvar imagens visualizadas atualmente, iniciar a gravação de vídeo (círculo vermelho), parar a gravação de vídeo (quadrado cinza), escolher o local onde as imagens e os vídeos serão salvos (Configuração) e finalizar o programa (Sair).

Figura 31 - Tela de execução do Projeto Controle Cliente

O software proposto, quando complementando com equipamentos robóticos, hardwares, placas de circuitos, motores, torna possível a construção de leques de possibilidades no ramo da Telerobótica. Dentre elas, foi escolhida a execução de Teleoperação de veículos automotores de quatro rodas, pela viabilidade de baixo custo, além de ser uma meta realizável a médio prazo. A Figura 32 ilustra a primeira versão do protótipo realizado.

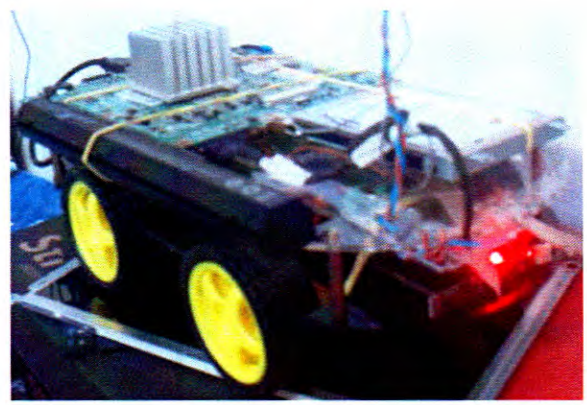

Figura 32 - Primeira versão do protótipo de Telerobótica: Projeto Controle

Percebeu-se que a partir da curiosidade de integrar as ações: captura de

imagens com a comunicação de computadores, atrelando isso à Robótica, surgiu à resolução de inúmeros problemas, tais como: desarmamento de bombas, sem que o militar corra risco de vida; explorar planetas, a partir da atmosfera, ou seja, sendo desnecessário o contato direto do astronauta com o planeta explorado, evitando riscos de vida, Telecirurgia, monitoramento de domicílio por meio da internet, dentre outros.

O sistema Telerobótico proposto é composto basicamente por duas partes, o ambiente operador e o ambiente remoto, estes conectados através de uma rede de computadores qualquer, seja ela privada ou não, sendo necessária uma velocidade de operação considerável, para que haja a mínima perda de tempo possível, esse ponto sendo primordial ao desempenho da execução.

O ambiente operador é controlado por um usuário com um conhecimento intermediário em rede de computadores indispensável para a configuração do mesmo, responsabilizando-se por enviar caracteres que representam o estado atual da vontade do usuário (executar alguma ação ou não) oriundo de um dispositivo de entrada como: teclado, *joypad*, *joystick*, dentre outros. Após o envio desse comando o ambiente operador, põe-se em modo de escuta, preparado para receber dados que representarão uma imagem capturada da câmera acoplada ao robô.

O ambiente remoto deve conter uma câmera, além de apresentar possibilidade de locomoção (rodas, esteiras, etc.), onde são capturadas imagens a serem enviadas ao ambiente operador podendo ser pré-processadas para equilibrar o nível de carga de processamento das informações em ambos os ambientes, exigindo um processador e uma memória primária, representados neste trabalho por um Netbook disponibilizando recursos essenciais como câmera, conexão com rede wireless, entre outros. Ao ser enviada a imagem, o robô espera por receber uma resposta de seu operador, que representa tanto uma resposta de que o dado foi recebido, quanto o comando a ser executado, formando assim, uma comunicação *half-duplex*[6] com o usuário.

Os ambientes, operador e remoto, devem manter-se conectados por uma velocidade de transmissão diretamente dependente do tamanho do quadro capturado, onde comprovado através de testes que a taxa de bits necessária para a transmissão de vídeo utilizando uma resolução de 320x240 pixels, mantém-se por volta de 1,5 Mbps, já se a resolução for estendida para 640x480, uma rede que proporcione 10 Mbps com facilidade

[6]Half-duplex é um tipo de comunicação no qual os dados podem fluir entre dois dispositivos, mas não simultaneamente. Cada dispositivo pode enviar e receber dados, mas apenas um dispositivo pode transmitir por vez.

será exigida para que o vídeo tenha desempenho satisfatório.

> O realismo da imagem está relacionado com nossa maneira de perceber ambientes, que é baseado em gradientes de textura, projeções de objetos, reflexos de luz, sombras e assim por diante. Características destes dados implicam a transmissão de imagens de alta qualidade que requerem uma grande largura de banda. [...] a largura de banda mínima é de 5,2 Mbps. Corresponde às imagens comuns compactados. A largura de banda máxima é de 344 Mbps, o que corresponde a imagens estereoscópicas com 70 quadros por segundo e uma resolução de 640x480 pixels por imagem. (ARACIL *et al*, 2007, traduzido pelo autor).

O Quadro 6 ilustra um comparativo entre qualidade das imagens associada a velocidades da conexão no meio de transmissão. Conforme os testes feitos, comprova-se que a taxa de bits necessária para a transmissão de vídeo, utilizando-se um codec de compressão DivX®, com a resolução de 320x240 pixels, mantém-se por volta de 1,5 Mbps, onde a conversão de cada imagem retorna um texto com tamanho exato de 230400 bytes. Contudo, se a resolução for estendida para 640x480 pixels, o tamanho do arquivo é de 921600 bytes, entretanto, para transmitir o vídeo com um desempenho satisfatório, é necessário uma rede que proporcione no mínimo 10 Mbps de velocidade em transferência de dados.

Quadro 6 - Quadro de velocidade da conexão por resolução de vídeo

Tipos de sinais	Tamanho do quadro	Frames por segundo	Largura de banda
TV vídeo (PAL/NTSC)	720x480	25 – 30 fps	165,9 Mbps
TV vídeo compactado (Qualidade de DVD)	720x480	25 – 30 fps	5,2 Mbps
Vídeo stereo (Não compactado)	640x480	30 – 70 fps	147 – 344 Mbps
Vídeo stereo compactado (Qualidade de DVD)	640x480	30 – 70 fps	6,3 – 14,6 Mbps

Logo abaixo, são exemplificados dois testes de desempenho da rede ao se transmitir imagens pela rede usando uma aplicação desenvolvida. Os equipamentos usados foram um computador Desktop Intel® Core 2 Duo 2.8Ghz com 2Gb de memória (Cliente), conectado a um roteador *Wireless* TP-LINK de 150 Mbps por um cabo de ethernet, conectado a um Netbook CCE Intel® Atom® 1.33Ghz com 2Gb de memória (Servidor), ambos os computadores com o Ubuntu 11.04 como sistema operacional.

Na Figura 33, em *Network History* do gerenciador do sistema do Ubuntu, é exibido um gráfico que representa uma velocidade estável em 1,5 Mbps na transmissão de imagens com resolução de 320x240 pixels usando a aplicação que originou este trabalho.

A estabilidade da velocidade da rede implica em um desempenho satisfatório, com uma taxa de frames por segundo (fps) próxima ao ideal, que quanto maior esta taxa, maior será a sensação de movimento que uma sequência de imagens pode propor e maior a quantidade de dados que serão enviados.

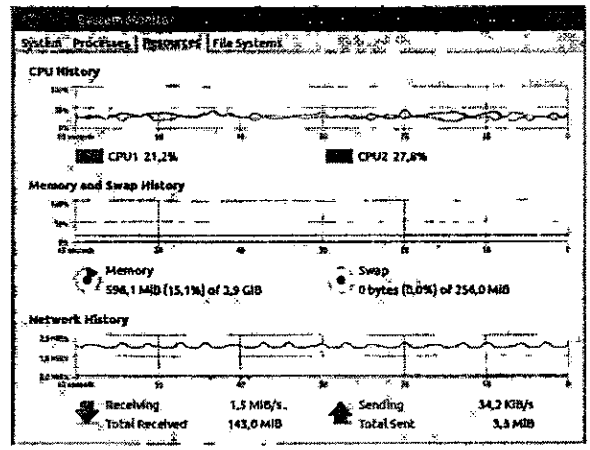

Figura 33 - *Network History* demonstra um gráfico do Cliente com a velocidade constante necessária para transmissão de imagens com resolução de 320x240 pixels

Uma possível instabilidade na rede pode ocasionar em perda significativa de desempenho na hora de se transmitir dados. A Figura 34 mostra, em *Network History*, irregularidades no gráfico de velocidade da rede. Este teste foi realizado usando as mesmas características do teste anterior: aplicação, resolução da imagem e equipamento de transmissão sem fio. Esta instabilidade foi propositalmente provocada ao se remover a antena do roteador *wireless*, resultando na perda do sinal, acarretando na demora na transmissão, e por fim, na redução da qualidade do vídeo.

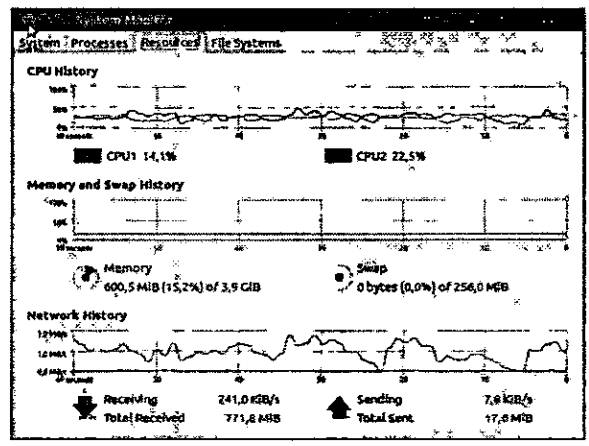

Figura 34 - *Network History* demonstra um gráfico do Cliente com a velocidade irregular, resultando em perda de desempenho do vídeo

Diferente do que muitos imaginam a possibilidade de se desenvolver um robô, mesmo que simples, não está a uma distância muito grande, a robótica está envolvida principalmente com placas e equipamentos físicos, e não diferente de software, existe hardware livre. Juntando-se estes dois componentes, torna-se possível a concretização de idéias que integradas a dedicação e incentivo, atingem seus objetivos.

Neste trabalho, são usados o Arduino integrado a um *shield* que possui a tarefa de controlar motores elétricos de corrente contínua (Figura 35), estes usados no presente estudo para controlar movimentos direcionais (simulando um veículo automotor de quatro rodas), permitindo dessa forma ao Arduino um controle de voltagens, bem maiores que a voltagem disponibilizada pelo próprio Arduino, necessárias para a alimentação dos motores.

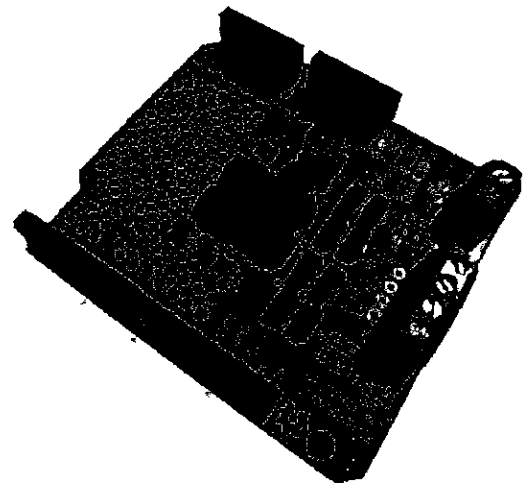

Figura 35 - *Shield* controlador de motores DC[7]

Em suma, utilizou-se o Arduino, com a finalidade de controlar os motores do experimento, no caso deste trabalho, atrelados a um *chassi* feito de acrílico de um carrinho que deve ter algumas peças de um Netbook necessárias para seu funcionamento como placa-mãe, HD e bateria, além do Arduino, o *shield* controlador dos motores, e uma fonte de energia para os motores independente da fonte do Netbook para serem capazes de executar o programa aqui proposto sendo possível, assim, sua execução onde essas peças estarão montadas em um chassi de acrílico como esquematizada na Figura 36.

Sabe-se que um equipamento computacional precisa de uma parte lógica para que seja executada a tarefa desejada, nesse sentido, destaca-se que o microchip do Arduino, pode ser programado com a linguagem C, esta que realiza em seguida, o processamento dos dados.

[7] DC - Corrente Contínua

Figura 36 - Chassi de acrílico

Por ser programável na linguagem C, o Arduino permite a execução paralela com outras linguagens de programação que têm essa mesma base. Como o núcleo do Python é desenvolvido também em C, acaba que por proporcionar uma interação, entre a linguagem e o Arduino, extremamente facilitada, para isso, é indispensável o módulo Serial que deve ser instalado junto ao Python por não vir em sua biblioteca padrão.

5 CONSIDERAÇÕES FINAIS

O objetivo do presente trabalho foi apresentar uma arquitetura para Telerobótica. Foram exibidos alguns equipamentos que podem ser usados com esse propósito, além da sugestão de softwares nas versões Cliente e Servidor desenvolvidos em Python, que podem ser adaptados a qualquer tipo de serviço que envolva o monitoramento através de câmeras. Para isso, dedicou-se grande parte do tempo na realização de testes a fim de comprovar o funcionamento aplicável do projeto.

No desenvolvimento deste trabalho, o maior empecilho foi a falta de embasamento teórico. Não foi achado qualquer material que mostre uma integração de todas as áreas abordadas desenvolvidas na linguagem Python. A partir desse pressuposto, decidiu-se abordar as partes envolvidas na proposta separadamente. Assim foi mostrado que essas partes trabalhando em conjunto, pode-se obter um sistema Telerobótico.

Devido à necessidade de equipamento de hardware na montagem do robô, também ocorreram dificuldades monetárias na aquisição das peças. Para se adquirir o equipamento com um preço menor, foi necessário comprar algumas das peças diretamente do exterior. Equipamentos de robótica comprados no país, podem chegar a valores de até cinco vezes maiores o valor do mesmo produto comprado de países como a China.

Após um ano de pesquisas e testes, a arquitetura proposta teve sua primeira versão funcional. Em geral, essa ideia teve bom resultados logo ao ser colocada em prática. A primeira versão do projeto obteve sucesso, mas ainda observou-se alguns detalhes que precisavam ser corrigidos.

O Netbook acoplado em cima do chassi de acrílico (Figura 32), mesmo sem sua carcaça, tornou-se muito pesado dificultando assim, o momento que o robô teria que realizar uma alteração de direção. Os pneus são feitos com uma borracha mole, aumentando o atrito entre os pneus e o chão, acarretando na dificuldade de se realizar uma curva. Deve ser levado em consideração que o chassi tem quatro pneus independentes, o que dificulta ainda mais a curva.

Ao se manter o Netbook desmontado, problemas como: falta de blindagem eletromagnética, perda de sinal da rede sem fio ou o risco de ter suas peças internas danificadas. Uma pessoa pode tocar o interior no Netbook e acarretar na queima de sua placa mãe por descarga de energia estática.

Além dos problemas mencionados anteriormente, observou-se uma instabilidade no momento em que o Python realiza comunicação serial com o Arduino.

Pode ocorrer a situação em que o Python envie um comando para o Arduino sem que este esteja preparado para receber o dado. Acarretando no travamento total do sistema. Sendo necessário que o programa tenha seu processo morto através do gerenciador de processos. De consciência desses problemas apresentados, decidiu-se como trabalhos futuros, a substituição no Netbook por uma placa chamada Raspberry PI Model B, onde esta possui o tamanho de um cartão de crédito com um processador e memória que podem se aproximar a capacidade de processamento do Netbook.

Percebeu-se, ainda, que a velocidade de rede necessária para a transmissão das imagens é relativamente alta. Outra alteração futura a ser feita é a implementação de um algoritmo de compressão de dados para diminuir o tamanho do dado a ser transmitido. Ocasionando, assim, uma grande redução de velocidade diminuindo a carga no meio de transmissão.Outro ponto importante a ser modificado é o fato das rodas dianteiras dificultarem a mudança de direção do carro, podendo serem substituídas por uma pequena roda giratória que exerceria menos atrito com o chão. A proposta apresentada, portanto, atingiu seu objetivo. Ocorreram pequenos problemas já mencionados, que podem ser alterados posteriormente, dando um melhor aperfeiçoamento dessa pesquisa.

REFERÊNCIAS

About Python. Python Software Foundation. Disponível em: <http://www.python.org/about/>. Acesso em: 23 mar. 2012.

ALCHIN, M. **Pro Python**. Berkeley: Apress, 2010.

ÁLVARES, A. J.; JUNIOR, L. S. J. R. **Telerobótica:** Metodologia para o desenvolvimento de sistemas robóticos teleoperados via Internet. Universidade de Brasília. Brasília. Disponível em: <ftp://ftp.graco.unb.br/pub/publicacoes_graco/aaacic.pdf>. Acesso em: 20 jun. 2012.

ANDERSON, A.; BENEDETTI, R. **Head First Networking**. Sebastopol: O'Reilly, 2009.

ARACIL, R. *et al.* The human role in telerobotics. **Advances in Telerobotics**, p. 11-24, 2007.

BARRY, P. **Head First Python**. Sebastopol: O'Reilly, 2011.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. Sebastopol: O'Reilly, 2008.

BRIGGS, A. **Hello! Python**. Shelter Island: Manning, 2012.

BUDD, T. A. **Exploring Python**. Maidenhead: McGraw-Hill Education EMEA, 2009.

BUSCH, J. **Origem da palavra Rede**. Disponível em: <http://jaderedes.blogspot.com.br/2008/11/origem-da-palavra-rede.html>. Acesso em: 01 abr. 2013.

CAVALCANTE, M. A. *et al.* **Física com Arduino para iniciantes**. Revista Brasileira de Ensino de Física. Disponível em: <http://www.scielo.br/pdf/rbef/v33n4/18.pdf>. Acesso em: 01 abr. 2013.

CHUN, W. J. **Core Python Programming**. 2.ed. Upper Saddle River: Prentice Hall, 2006.

COSTA, Maria Nicolaci da. **Revoluções Tecnológicas e Transformações Subjetivas**. Rio de Janeiro, 2002.

DARAYA, V. **Curiosity pode ter feito descoberta histórica em Marte**. Disponível em: <http://info.abril.com.br/noticias/ciencia/curiosity-pode-ter-feito-descoberta-historica-em-marte-21112012-42.shl>. Acesso em: 30 mar. 2013.

DEMAAGD, K. *et al.* **Practical Computer Vision with SimpleCV**. Sebastopol: O'Reilly, 2012.

DIAS, C.; HOUNSELL, M. da S.; JUNIOR, R. S. U. R. Análise de Colaboratividade Usando Telerobótica. **International Conference on Engineering and Technology Education**. Ilhéus, p. 543, março. 2010.

HELLMANN, D. **The Python Standard Library by Example**. Boston: Pearson

Education, 2011.

KEEL, S. A. **What Is an Internet Socket?**. Disponível em: <http://www.wisegeek.com/what-is-an-internet-socket.htm>. Acesso em: 01 abr. 2013.

LABAKI, J. **Pensando em Tkinter**. Universidade Estadual Paulista. Ilha Solteira. Disponível em: <http://www.fem.unicamp.br/~labaki/Python/PensandoTkinter.pdf>. Acesso em: 15 mar. 2012.

LAGANIÈRE, R. **OpenCV 2 Computer Vision Application Programming Cookbook**. Birmingham: Packt Publishing, 2011.

LUTZ, M. **Learning Python**. 4.ed. Sebastopol: O'Reilly, 2009.

LUTZ, M. **Programming Python**. 4.ed. Sebastopol: O'Reilly, 2011.

MARGOLIS, M. **Arduino Cookbook**. 2.ed. Sebastopol: O'Reilly, 2012.

MARQUES, A. L. *et al.* **Web Controlled Robotic Arm.** Instituto Superior de Engenharia de Coimbra. Coimbra. Disponível em: <http://www.aedie.org/9CHLIE-paper-send/381-MARQUES.pdf>. Acesso em: 08 out. 2012.

MARTINEZ, M. *et al.* **UML**. Disponível em: <http://www.infoescola.com/engenharia-de-software/uml/>. Acesso em: 02 abr. 2013.

MARTINS, A. **O que é Robótica**. São Paulo: Editora Brasiliense, 2006.

MCGUGAN, W. **Beginning Game Development with Python and Pygame**: From Novice to Professional. Berkeley: Apress, 2007.

MENÉNDEZ, P. G. Las estrategias de resolución de problemas y el estúdio científico del riesgo: el caso de los alimentos transgênicos. In: LUJÁN, J. L. y ECHEVERRÍA, Jr. **Gobernar los Riesgos: ciencia y valores en la sociedad del riesgo.** Madrid: Biblioteca Nueva – OEI, 2004, pp. 263-287.

MILES, R.; HAMILTON, K. **Learning UML 2.0**. Sebastopol: O'Reilly, 2006.

PAYNE, J. **Beginning Python**: Using Python 2.6 and 3.1. Indianapolis: Wrox, 2010.

PINHEIRO, J. M. S. **Quem é o Profissional de Telemática**. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_quem_eh_o_profissional_de_telematica.php>. Acesso em: 30 mar. 2013.

Python 2.7 License. Python Software Foundation. Disponível em: <http://www.python.org/getit/releases/2.7/license/>. Acesso em: 23 mar. 2012.

Python Imaging Library (PIL). Disponível em: <http://www.pythonware.com/media/data/pil-handbook.pdf>. Acesso em: 23 mar. 2012.

REBOUCAS, F. **Curiosity**. Disponível em: <http://www.infoescola.com/exploracao-

espacial/curiosity/>. Acesso em: 30 mar. 2013.

RHODES, B.; GOERZEN, J. **Foundations of Python Network Programming**: The comprehensive guide to building network applications with Python. 2.ed. Berkeley: Apress, 2010.

SILEIKA, R. **Pro Python System Administration**. Berkeley: Apress, 2010.

TANENBAUM, A. **Redes de Computadores**. Tradução: Vandenberg D. de Souza. 4. ed. Rio de Janeiro: Elsevier, 2003.

THORNE, B.; GRASSET, R. **Python for Prototyping Computer Vision Applications**. University of Canterbury. Christchurch. Disponível em: <http://ir.canterbury.ac.nz/bitstream/10092/5430/1/12630971_2010-Python_for_Prototyping_Computer_Vision_Applications.pdf>. Acesso em: 15 nov. 2011.

WANGENHEIM, A. von **Introdução à Visão Computacional**. Disponível em: <http://www.inf.ufsc.br/~visao/>. Acesso em: 01 abr. 2013.