UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

E DE COMPUTAÇÃO

# SEL0630 - Aplicações de Microprocessadores II

# R3V - *Remote 3D Viewer*

**Autor(es)**: Davi Diório Mendes 7546989

Henrique Alberto Rusa 7593714

**Professor**: Evandro Luis Linhari Rodrigues

2015

# Resumo

O R3V consiste de um sistema distruibuido entre uma aplicação Android e um servidor, implementado como um *webservice* REST em uma placa de desenvolvimento Intel Galileo. O servidor provê serviços sobre o protocolo *HTTP* visando controlar uma câmera IP através dos movimentos de um *smartphone* Android dentro de um Google Cardboard, transportando assim o usuário para outro ambiente. Através do conceito de *dead reckoning*, rastreia-se o posicionamento real angular da câmera IP e, através de uma transformação matemática discretiza-se os movimentos para se compor a atuação de acordo com os comandos da aplicação cliente.

Palavras-Chave: Android, Galileo, Cardboard, *dead-reckoning*, REST, Flask.

# Sumário

# Capítulo 1

# Introdução

Neste documento apresenta-se uma possível abordagem do desenvolvimento de um protótipo de sistemas distribuídos para realidade virtual, onde transportamos o usuário para outro ambiente. Para isso controla-se uma câmera remotamente através dos sensores inerciais de um *smartphone* Android. O projeto se baseia no desenvolvimento de dispositivos semelhantes, como por exemplo o Oculus Rift que propõe a criação de uma nova plataforma de interação para o usuário. Seguindo este conceito, uma abordagem mais simples foi proposta, utilizando-se do Cardboard da Google que constrói, utilizando-se de uma estrutura em papelão e o celular do usuário, um óculos de realidade virtual.

## 1.1 Objetivos

A partir deste projeto pretende-se desenvolver um sistema de monitoramento em tempo real onde o usuário tem o controle, através de seu *smartphone*, de uma câmera IP. A aplicação se utiliza-se de realidade aumentada, simulando a presença do observador no ambiente observado.
O efeito de visão tridimensinal é dado através do algoritmo de Parallax que se baseia numa imagem 2D e constrói duas imagens, que compòe uma simulação 3D.

## 1.2 Organização do Trabalho(opcional)

Este trabalho está distribuído em 5 capítulos, incluindo esta introdução. Neste capítulo intrduziu-se o projeto a ser descrito neste documento. O segundo capítulo apresenta os conceitos necessários para o entendimento da teoria utilizada. O terceiro capítulo discorre sobre os materiais e métodos necessários para o desenvolvimento do projetio. O quarto capítulo apresenta os resultados obtidos com o projeto.

Por fim, no quinto capítulo, tiram-se conclusões do projeto apresentado neste documento.

# Capítulo 2

# Embasamento Teórico

Na fase de projeto do sistema R3V proposto foram elencados conhecimentos e fundamentos necessários para o desenvolvimento do mesmo. Com isso em mente, pode-se discursar melhor sobre os tópicos mais relevantes do sistema, permitindo que o leitor aprofunde-se devidamente para que, ao final, os objetivos do projeto sejam melhor discutidos e compreendidos.

Neste capítulo pretende-se analisar os conceitos de estereoscopia, sistemas embarcados, sistemas distribuídos e *dead reckoning*.

## 2.1 Estereoscopia

A simulação de imagens em três dimensões (visão espacial) é associada ao conceito da estereoscopia. Para isso, o cérebro humano necessita adiquirir duas imagens, cada qual com um leve deslocamento lateral (angular), e com estas calcular a profundidade dos objetos [1] [2].

Existem também técnicas de percepção do espaço tri-dimensional que avaliam sombras, sobreposição de objetos numa cena e vários outros parâmetros, possibilitando verificar a disposição dos elementos no espaço.

Entretanto, existe uma diferença fundamental entre percepção e simulação do espaço tri-dimensioanl para o usuário, apresentadas a seguir.

- **Percepção Tri-dimensonal:** O usuário consegue perceber a disposição dos elementos numa imagem (profundidade relativa, distância relativa entre objetos) somente a partir da análise de uma imagem que possui indicativos como sombra, iluminação e outros.

- **Simulação Tri-dimensonal:** O usuário tem a sensação de que os objetos possuem dimensões reais, com tamanho e profundidade bem definidos; a distância entre elementos é percebida,

entretanto não mais pela sombra, mas pela própria projeção da simulação em si.

Portanto, enquanto o usuário pode perceber a terceira dimensão em valores relativos (objetos mais próximos ou mais afastados, maiores ou menores), a simulação do espaço tri-dimensioanl é realizada pela sobreposição de imagens com mesmo ponto focal mas deslocadas levemente uma da outra, possibilitando a reconstrução, pelo cérebro, da profundidade dos elementos.

## 2.2 Sistemas Embarcados

Segundo White (2011, p.1) a definição de sistemas embarcados varia de pessoas para pessoas. Para alguém acostumado a lidar com servidores, programação para dispositivos móveis pode ser considerada como um desenvolvimento embarcado. Por outro lado, para aquele acostumado a trabalhar com microcontroladores de 8-bits, qualquer coisa com um sistema operacional não parece muito embarcado. Mas ela enuncia a definição: "um sistema embarcado é um sistema computadorizado construído propositadamente para sua aplicação" [3].

Desta forma podemos realçar a diferença entre sistemas computacionais e sistemas embarcados. Enquanto sistemas computacionas são desenvolvidos para atuar em computadores de propósito geral, sistemas embarcados são desenvolvidos de forma a serem embarcados em *hardware* com propósitos específicos, visando a aplicação.

## 2.3 Sistemas Distribuídos

Sistemas distribuídos é algo de difícil definição. Cada autor utiliza uma definição diferente, embora todas remetam que há computadores, ou outro dispositivo com capacidade de processamento, trabalhando em conjunto. Vamos tomar por definição: "Um sistema distribuído é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens." [4].

## 2.4 *WebServices REST*

Hoje a internet não é mais simplesmente uma ferramenta de difusão de informação. Diversos serviços são prestados através dos *Web Services*. Normalmente envolvem um processo burocrático rigoroso, como requisições SOAP ou outros tipos de mensagem, conforme o protocolo seguido. A idéia de um *WebService* REST é de prover serviços diretamente sobre o protocolo HTTP, sem a necessidade de adicionar uma camada de protocolo de serviço [5]. Serviços *REST* estão se popularizando cada vez mais, dada a baixa complexidade do sistema e fácil integração por parte dos usuários.

## 2.5 *Dead Reckoning*

A tecnologia provê diversos sistemas de posicionamento por sensores, *beacons* (referências fixas no espaço), equivalência entre mapas e outros. Entretanto, pode-se distinguir duas metodologias utilizadas: posicionamento relativo e absoluto [6].

- **Absoluto:** Se vale de técnicas de sensoreamento com referências fixas, previamente implementadas, o que adiciona um custo muito alto para cosntrução e manutenção destes sistemas espalhados pelo terreno em questão.
- **Relativo:** Propõe uma formulação mais elegante, se valendo de sensores e parâmetros intrínsecos da implementação. Com isso, pode-se estimar a posição atual do elemento de acordo com movimentações perceptíveis ao sistema.

*Dead reckoning* é uma implementação de posicionamento relativo que permite o programador avaliar os diversos sensores para se estimar a localização entre os períodos de tempos avaliados.

Note que as duas implementações de posicionamento possuem suas vantagens e desvantagens, sendo que uma estimativa de localização (posição relativa) adiciona erros às medidas, enquanto que a outra possui as referências extremamente precisas (posição absoluta).

## 2.6 Rotação dos Eixos

A decomposição dos movimentos de rotação de um usuário utilizando um óculos de virtualização é realizada com movimentos angulares, descritas pelos valores *pitch*, *yaw* e *roll*.
A figura 2.1 apresenta a orientação das rotações descritas anteriormente.

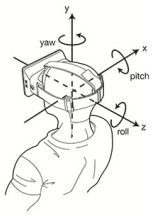


Figura 2.1: Representação dos movimentos de visão de um usuário (visto em: https://s3.amazonaws.com/static.oculus.com/website/2013/05/oculus_head_model.jpg)

# Capítulo 3

# Materiais e Métodos

Neste seção irá se discursar sobre os materiais e métodos utilizados no desenvolvimento do projeto *R3V*. Os materiais englobam todos os dispositivos físicos e produtos lógicos (bibliotecas de código, etc) utilizados no desenvolvimento do projeto proposto. Os métodos apresentam como e por meio de qual técnica alguns processos do sistema foram desenvolvidos, abstraindo o produto e focando na maneira com que foi implementado.

## 3.1 Materiais

A seguir são apresentados os materiais relevantes para o projeto proposto (*R3V*), sendo eles: placa de desenvolvimento Intel Galileo, *smarthphone Android*, IPCam (câmera controlada por rede) e Flask (servidor python).

### 3.1.1 Intel Galileo

A placa de desenvolvimento Intel® Galileo (figura 3.1), projetada e vendida pela Intel®, foi criada com base no processador Intel® Quark SoC X1000 de aplicação com o intuito de ser compatível com os *shields* de Arduino.

O processador provê para o usuário final uma arquitetura de 32 bits e *clock* de 400 MHz, conectores Ethernet 10/100, PCI-Express *mini-card*, USB 2.0, USB cliente (usado para programação) e botões de *reboot* e *reset*.

Como apresentado anteriormente, a placa possui diversas interfaces de conexão para viabilizar a comunicação com um computador, outro Arduino e até outros microcontroladores.

A Galileo suporta programação através da IDE do Arduino como também a utilização de um sistema operacional embarcado instalado em seu *hardware*.

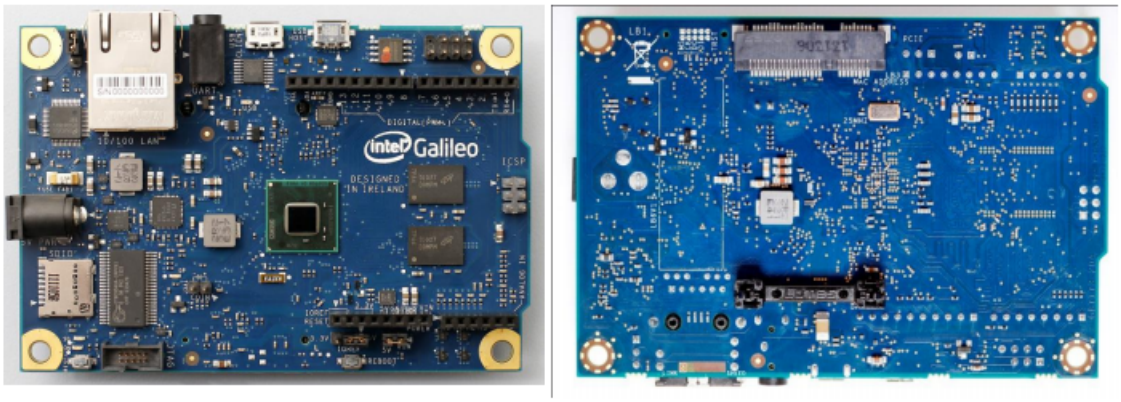


Figura 3.1: Imagem da placa de desenvolvimento Galileo - visão frontal (*Galileo Front*), visão de trás (*Galileo Back*) (visto em: http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-g1-datasheet.html)

A sua aplicação dentro do projeto desenvolvido é prover um *webservice* REST através do *framework* Flask.

### 3.1.2 Smartphone Android

Atualmente, os *smartphones* Android vem sendo uma ótima plataforma para desenvolvimento. Já integrado com diversos sensores e dispositivos de entrada e saída; como acelerômetro, GPS, câmera, microfone, sistema de som, tela, entre outros; além de disponibilizar uma sólida ferramenta de desenvolvimento, a Android SDK.

Dos *hardwares* providos pelo *smartphone* foi utilizado na aplicação os sensores inerciais, a fim de obter os angulos de euler da orientação do dispositivo; a tela, para a exibição da interface gráfica e a conexão *wifi*, para comunicação com a Intel Galileo. A forma como estes recursos foram utilizados será abordada na exposição da aplicação Android.

### 3.1.3 IPCam

Para o *streaming* de video, uma câmera IP foi utilizada afim de disponibilizar um serviço em rede e desacoplar o dispositivo do servidor (placa de desenvolvimento Intel® Galileo). Este *hardware* provê, através de um protocolo *http*, um serviço que executa diversas operações através de *scripts* CGI.

As intertfaces CGI da câmera apresentam três níveis diferenciados de permissão, sendo eles: visitante, operador e administrador. A funcionalidade utilizada pelo projeto somente requisitou acesso de operador (usuário e senha fornecidos pelo professor).

Além de serviços de video, a câmera em questão possui *scripts* para monitoramento, detecção de movimento, movimentação da mesma e diversos outros.

O projeto R3V utilizou-se de dois *scripts*, chamados: *decoder_control.cgi* e *videostream.cgi*.

O *script videostream.cgi* realiza a captura e envio do video através da rede para o cliente, enquanto que o *decoder_control.cgi* controla a movimentação da câmera em sua base. Abaixo (tabela 3.1) [7] se destaca os comandos de movimentação que foram utilizados no projeto.

Tabela 3.1: Comandos utilizados no projeto R3V da câmera IPCam FOSCAM

| Comando | Descrição |
|---|---|
| 0 | Move a câmera para baixo |
| 2 | Move a câmera para cima |
| 4 | Move a câmera para a esquerda |
| 6 | Move a câmera para a direita |
| 25 | Coloca a câmera no seu centro (implementação própria) |
| 90 | Move a câmera para a diagonal esquerda inferior |
| 91 | Move a câmera para a diagonal direita inferior |
| 92 | Move a câmera para a digonal esquerda superior |
| 93 | Move a câmera para a diagonal direita superior |

Para mais informações sobre o funcionamento da câmera utilizada consulte o manual do fabricante: [7].

### 3.1.4 Flask

"O Flask é um microframework para python, baseado no *Werkzeug*, *Jinja2* e boas intenções". Com o *Flask* podemos facilmente criar um *backend* para processar as requisições à Intel Galileo, implementando assim um *WebService* REST.

Para utilizarmos o Flask, atrelamos um método python a uma determinada URL. Assim conseguimos recuperar dados transmitidos através da requisição HTTP, processá-los e retornarmos uma resposta. Através do Flask, o R3V provê três serviços: *streaming* de vídeo de uma câmera IP, movimentação desta câmera e *dead reckoning* do movimento da câmera. A seguir abordamos melhor o provisionamento de cada um destes serviços.

Através da URL <*endereço_da_intel_galileo*>/camstream/ provemos o serviço de *streaming* de vídeo. Este endereço retorna um *stream* MJPEG ouseja, uma sequência de imagens JPEG. Este serviço recupera as imagens da câmera pelo mesmo serviço de MJPEG. Desta forma ele atua somente como um *proxy*, encapsulando o serviço de *stream* de vídeo da própria câmera IP.

Através da URL <*endereço_da_intel_galileo*>/camposition/cam_step/?move=<direção> provemos o serviço de movimentação da câmera IP em passos de cinco graus. No parâmatro direção deve-se informar o código da direção que se deve atuar. As direções são mapeadas conforme a tabela 3.2.

Tabela 3.2: Código de direção para requisitar movimentação da câmera.

| Código | Direção |
|---|---|
| 0 | abaixo |
| 2 | acima |
| 4 | esquerda |
| 6 | direita |
| 90 | diagonal esquerda abaixo |
| 91 | diagonal direita abaixo |
| 92 | diagonal esquerda acima |
| 93 | diagonal direita acima |

Através da URL <*endereço_da_intel_galileo*>/camposition/set_zero/ inicializamos o sistema de *dead reckoning* do movimento da câmera, tomando a posição atual da câmera como zero graus de rotação em torno do eixo *x* e zero graus de rotação em torno do eixo *y*.

Através da URL <*endereço_da_intel_galileo*>/camposition/?pitch=<x>&yaw=<y> provemos o serviço de movimentação da câmera com *dead reckoning* do movimento. No parâmetro *x* deve-se atribuir qual a posição angular desejada em torno do eixo *x*. No parâmetro *y* deve-se atribuir qual a posição angular desejada em torno do eixo *y*. O serviço de movimentação por *dead reckoning* só aceita posições angulares para *pitch* entre 80 e -30 graus e posições angulares para *yaw* entre 100 e -100 graus.

Desta forma temos a Intel Galileo provendo os serviçõe de *streaming*, *movimentação* e *dead reckoning* da posição de uma câmera IP.

## 3.2 Métodos

No desenvolvimento de um projeto várias técnicas são utilizadas para implementar diversos componentes necessários para o correto funcionamento do sistema, tendo em vista eficiência e qualidade no produto final.

Nesta seção serão apresentados todos os métodos, implementações e comportamentos dos componentes desenvolvidos para se construir o sistema R3V. Com isso, irá se discursar um pouco sobre

*streaming* de video (MJPEG), mapeamento de movimento (*dead reckoning*) e o servidor desenvolvido.

### 3.2.1 MJPEG

### 3.2.2 *Dead Reckoning*

Uma das estruturas fundamentais do comportamento geral do sistema R3V se compõe do mapeamento e controle dos movimentos do usuário, afim de proporcionar uma fluidez na movimentação da câmera no lado da IPCam.

Utilizando os conceitos de *dead reckoning* (posicionamento de acordo com leitura de sensores e estimativa de movimento), uma transformação matemática foi aplicada aos resultados do sensor inercial (acelerômetro) adiquiridos através do aplicativo do *smartphone*, resultando em um discretização do espaço de visão do usuário. Isso se deve ao fato de que a IPCam possui somente movimentos bem definidos e não responde a movimentos fora de seus comandos pré-definidos.

Através do SDK do *Cardboard* da Google® foi possível adiquirir os movimentos angulares denominados *pitch* e *yaw*, que correspondem a movimentos em torno de um eixo bem definido (eixo $x$ e $y$ respectivamente).

A transformação realizou a conversão do movimento para um valor da superfície de uma esfera (quadrante), por meio de uma função trigonométrica. Assim

$$arctan2(x,y) = \begin{cases} arctan(\frac{y}{x}) & x > 0 \\ arctan(\frac{y}{x}) + \pi & x < 0 \; and \; y \geq 0 \\ arctan(\frac{y}{x}) - \pi & x < 0 \; and \; y < 0 \\ \frac{\pi}{2} & x = 0 \; and \; y > 0 \\ -\frac{\pi}{2} & x = 0 \; and \; y < 0 \\ undefined & x = 0 \; and \; y = 0 \end{cases}$$

O resultado de $arctan2(x,y)$ retorna um valor entre $[-\pi,\pi]$. Este valor apresenta o quadrante o qual a visão do usuário está olhando no momento. Portanto, foi possível discretizar o espaço do campo de visão em segmentos, onde o mapeamento da visão possibilita, através dos comandos padrões da câmera IPCam, simular um movimento natural do usuário no dispositivo ligado em rede.

Uma ressalva a se fazer a este método é que o sistema necessita armazenar o valor da última posição para que se possa realizar o cálculo do deslocamento angular da câmera. Com isso, foi possível se rastrear e mapear o campo de visão com o movimento do usuário.

### 3.2.3 Aplicação Cliente

A aplicação cliente foi desenvolvida como um aplicativo Android. Para o desenvolvimento de aplicativos Android é feito o uso do Android SDK. Ele provê meios de criar interfaces gráficas, aplicativos com processamento paralelo, comunicação interprocesso, conexão com servidores HTTP, e muitas outras funcionalidades. Neste projeto nos interessa a criação de uma interface gráfica para utilizar o *smartphone* no Google Cardboard, o acesso aos sensores inerciais (acelerômetro, giroscópio e magnetômetro) e a comunicação com servidores HTTP.

Para a interface gráfica, o necessário é podermos exibir uma imagem em cada metade da tela, com o celular em orientação paisagem. Ambas as imagens deve ser a mesma, de forma a dar o aspecto tridimensional quando no *cardboard*. Para nos beneficiarmos de algumas facilidades do *Cardboard SDK*, discutidas adiante, foi necessário manter a interface de renderizador gráfico do *Cardboard*, mas uma nova interface foi sobreposta sobre a padrão.

O acesso aos sensores inerciais foi feito indiretamente através do *Cardboard SDK*. Ela abstrai o cálculo dos ângulos de euler do dispositivo como uma requisição da direção de visão do usuário. Desta forma temos os ângulos ao redor dos eixos $x$ e $y$ sem termos que diretamente acessar os sensores inerciais e tratar suas leituras.

Outro recurso do Android utilizado foi a comunicação com servidores HTTP. A comunicação entre o *smartphone* Android e a Intel Galileo caracteriza o R3V como um sistema distribuído. Diz-se isto pois possuímos dois dispositivos computacionais, conectados pela internet e operando em conjunto através de troca de mensagens. Com informação trocada entre os dispositivos temos o *streaming* de vídeo da Galileo para o *smartphone* e o pedido de movimentação da câmera, do dispositivo móvel para o servidor, requisitando o mesmo posicionamento angular lado no cliente.

Na recepção do *streaming* de vídeo foi utilizada a classe MjpegInputStream. A classe foi reutilizada do projeto arduino-android-blueprints, de Marco Schwartz, mas a autoria da classe é desconhecida e esta é utilizada em diversos projetos.

No comando de movimentação da câmera há dois tratamentos para evitar o uso abusivo da rede. Um dos tratamentos é por segmentação de movimento. Só é enviado um comando, do *smartphone* para o servidor, se o movimento tiver pelo menos três graus de variação seja no eixo $x$ ou no eixo $y$. Outro método de redução de uso da rede é nunca enviar um comando com menos de 200 milisegundos de espaçamento do comando anterior. Desta forma reduzimos o uso de dados do dispositivo celular e evitamos de sobrecarregar o servidor com muitas requisições. Outro contato do dispositivo móvel com a Intel Galileo é na inicialização do aplicativo, onde requisita-se que o servidor zere as posições angulares da câmera, a fim de sincronizar o movimento do *smartphone* com a câmera IP.

# Capítulo 4

# Resultados e Discussões

Primeiramente observou-se que a configuração da Intel Galileo, além de possuir pouca documentação, apresenta alguns erros, dado que o suporte à primeira geração do dispositivo foi descontinuado. Com isso, o *overhead* de trabalho para início do projeto foi afetado, ocasionando um buraco na programação do grupo.

No decorrer do desenvolvimento da aplicação o grupo foi impossibilitado de dar continuidade aos trabalhos dado que a imagem do sistema operacional embarcado possuia falhas críticas no OpenCV para a linguagem python (que foi requisitada inicialmente como parte do projeto). Uma solução foi apresentada meses depois com a utilização de uma nova distribuição de sistema operacional, já com as correções necessárias. Uma consequência deste atraso citado é a não utilização do algoritmo de Parallax para estimação de imagens 3D.

O servidor *web* baseado no *framework* Flask trouxe muita facilidade no manuseio das requisições do *backend* criadas para a comunicação dos serviços da câmera IP com a aplicação Android, já que possuia uma arquitetura montada para gerênciar os recursos de rede.

A aplicação Android foi construída por cima de um código de uma aplicação Cardboard. Esta biblioteca apresenta aos usuários diversas facilidades, como a obtenção dos ângulos de Euler de rotação do sensor inercial do *smartphone*, facilitando o trabalho do grupo na utilização das respostas dos sensores. A aplicação final apresenta um *streaming* de video de baixa latência com qualidade de imagem em $640x480$ *pixels*, com possível configuração através de comandos próprios da câmera IP.

O mecânismo de *dead reckoning* na movimentaçào da câmera apresenta erros desprezíveis proporcionando uma boa expoeriência ao usuário.

# Capítulo 5

# Conclusão

Durante o desenvolvimento de projeto os objetivos foram alcançados com qualidade superior à esperada. O único ponto que ficou em falta foi o desenvolvimento do algoritmo de Parallax para estimar imagens 3D baseado em uma única imagem 2D.

## Trabalhos futuros

O protótipo se encontra numa fase inicial de desenvolvimento, onde muito pode se fazer para otimizar e aperfeiçoar os processos e serviços utilizados.
Com isso, propõe-se a criação de uma câmera atrelada à placa de desenvolvimento Galileo onde se utilizaria motores mais precisos e com controle individual para se realizar movimentos mais naturais e responsivos.
Outra observação válida a ser feita é na implementação do algoritmo de Parallax que serviria de fonte para a simulaç!ao de um espaço tridimensional para o usuário.

# Referências Bibliográficas

[1] GUSTAVO RP ESTEVES, MA FEITOSA, and Bruno José Torres Fernandes. Estereoscopia no cálculo de distância e controle de plataforma robótica. *UPE-Universidade de Pernambuco, Pernambuco, RE*, 2011.

[2] Edson Momm. Protótipo de um ambiente de visualização com técnicas de estereoscopia. *Trabalho de conclusão de curso de Ciências da Computação. Orientador Prof. Dalton Solano dos Reis. Universidade Regional de Blumenau. Blumenau*, 2001.

[3] Elecia White. *Making Embedded Systems: Design Patterns for Great Software*. O'Reilly Media, Inc., 2011. ISBN 1449302149, 9781449302146.

[4] G. COULOURIS, J. DOLLIMORE, and TIM KINDBERG. *Sistemas Distribuídos - 4ed: Conceitos e Projeto*. BOOKMAN COMPANHIA ED, 2007. ISBN 9788560031498. URL `https://books.google.com.br/books?id=KSZ1rIRWmUoC`.

[5] Tharcis Dal Moro, Carina Dorneles, and Marcelo Trindade Rebonatto. Web services ws-* versus web services rest. *Revista de Iniciação Científica*, 11(1), 2009.

[6] Johann Borenstein and Liqiang Feng. Umbmark: a method for measuring, comparing, and correcting dead-reckoning errors in mobile robots. 1994.

[7] Shenzhen Foscam Technology. *IP Camera CGI*. 37 pp.