

Daniel Noriaki Kurosawa

**Imersão em ambiente remoto através da
operação de braço robótico por rastreamento
de movimentos sem marcadores e sistema de
câmeras estereoscópicas**

São Paulo

2016

Daniel Noriaki Kurosawa

**Imersão em ambiente remoto através da operação de
braço robótico por rastreamento de movimentos sem
marcadores e sistema de câmeras estereoscópicas**

Trabalho de Formatura apresentado ao Departamento de Engenharia de Telecomunicações e Controle da Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro

Universidade de São Paulo - USP
Escola Politécnica
Engenharia Elétrica com ênfase em Automação e Controle

São Paulo
2016

Daniel Noriaki Kurosawa

Imersão em ambiente remoto através da operação de braço robótico por rastreamento de movimentos sem marcadores e sistema de câmeras estereoscópicas/ Daniel Noriaki Kurosawa. – São Paulo, 2016-

64 p. : il. (algumas color.) ; 30 cm.

Trabalho de Conclusão de Curso – Universidade de São Paulo - USP
Escola Politécnica

Engenharia Elétrica com ênfase em Automação e Controle, 2016.

1. Engenharia 2. Engenharia de Automação e Controle I. Universidade de São Paulo. Escola Politécnica. Departamento de Telecomunicações e Controle

Agradecimentos

Agradeço imensamente:

À instituição de ensino e todos os seus docentes que não só contribuiram para minha formação acadêmica, mas que através de lições e ensinamentos, contribuiram para minha formação como cidadão.

Ao orientador, Prof. Dr. Reginaldo Arakaki e ao co-orientador Prof. Dr Fuad Kassab Junior, por todo o apoio e paciência. O bom humor e a confiança depositados foram grande fonte de motivação durante o decorrer deste trabalho.

Aos engenheiros Marcelo Angelo Pita e Leandro Rodrigues de Souza, por toda orientação e o auxílio prestado na implementação do projeto.

À Camila Aya Uratsuka, por contribuir com as figuras que ilustram este trabalho.

À André Hashimoto Oku, por fornecer hardware usado durante a fase de testes deste projeto.

À minha família pelo amor, incentivo e apoio incondicional.

Por fim aos amigos, pelos constantes incentivos.

*“The only difference between screwing around and science ,
...is writing it down.”*
(Adam Savage, Mythbusters)

Resumo

Este trabalho estuda a interação imersiva em um ambiente remoto através do uso de estereoscopia e de operação ótica sem marcadores de um braço robótico.

Para tal objetivo, foi definido um projeto cuja execução gerou um sistema composto de:

- Microsoft Kinect, que envia os dados capturados a um braço robótico através de um servidor.
- Um sistema de câmeras estereoscópicas que envia suas imagens a um servidor, que podem ser acessadas remotamente por um óculos de realidade virtual ou qualquer dispositivo que tenha um browser compatível instalado.
- Um sistema de captura de movimentos da cabeça do usuário via giroscópio, que envia estes dados remotamente a um conjunto de motores localizados na base da câmera, de modo a sincronizar os movimentos da cabeça do usuário a visão das câmeras.

Os resultados finais indicam um grande espaço para o desenvolvimento de projetos deste tipo em aplicações futuras, apesar da necessidade de melhora de precisão para usos comerciais.

Palavras-chave: Internet of Things. Microsoft Kinect. Estereoscopia. Braço Robótico. IoT.

Abstract

This work studies the immersive interaction with a remote environment by using stereoscopy and markerless operation of a robotic arm.

Para tal objetivo, foi definido um projeto cuja execução gerou um sistema composto de: which create a system that comprises:

- A Microsoft Kinect, which sends captured data to a robot controller via server.
- A stereoscopic camera system, which sends its images to a server, which are in turn accessed by VR Glasses or any device with a compatible browser installed.
- A gyroscope based head-tracking system, which sends user's head pan/tilt data to motors located in the camera system's base, allowing the cameras to move in synchronization with the user's head movements.

Final results indicate great applicability for this approach in future uses. In its current state however, this approach lacks precision necessary for commercial use.

Keywords: Internet of Things. Microsoft Kinect. Stereoscopy. Robotic Arm. IoT.

Lista de ilustrações

Figura 1 – Diagrama do projeto	24
Figura 2 – Elementos de informação do sistema	28
Figura 3 – Manipulação de dados pelo sistema do braço robótico	29
Figura 4 – Manipulação de dados pelo sistema de câmeras estereoscópicas	30
Figura 5 – Manipulação de dados pelo sistema de motores da câmera	30
Figura 6 – Tecnologias Utilizadas para Conexão entre os Nós	31
Figura 7 – Arquitetura do sistema de operação do braço robótico	35
Figura 8 – Arquitetura do sistema de motores da câmera	36
Figura 9 – Arquitetura do sistema de captura de imagens estereoscópicas	37
Figura 10 – Visão tecnológica de Hardware do sistema	37
Figura 11 – Logotipo Kinect	38
Figura 12 – Kinect 2.0	38
Figura 13 – Logotipo Particle	39
Figura 14 – Particle Photon	39
Figura 15 – Logotipo Raspberry Pi	41
Figura 16 – Raspberry Pi Modelo B versão 3	41
Figura 17 – Logo da linguagem Java	42
Figura 18 – Logo Apache HTTP Server	45
Figura 19 – Diagrama de Gantt do sistema	47
Figura 20 – Kanban do projeto usando Trello	48
Figura 21 – Visual Studio 2015 IDE	49
Figura 22 – Editor de texto GNU NANO do Raspberry Pi	50
Figura 23 – Particle IDE	51
Figura 24 – Garra do braço robótico	53
Figura 25 – Coordenadas X-Y para todos os valores de theta1 e theta2	54
Figura 26 – Coordenadas X-Y-Z para todos os valores de theta1, theta2 e theta3	54
Figura 27 – Coordenadas X-Y-Z para todos os valores de theta1, theta2, theta3 e theta4	55
Figura 28 – Imagem obtida pelas câmeras estereoscópicas usando mjpg-streamer	56
Figura 29 – Montagem das câmeras estereoscópicas	56
Figura 30 – Óculos de realidade virtual com destaque para o giroscópio externo	57
Figura 31 – Imagem obtida pelas câmeras estereoscópicas	58
Figura 32 – Captura de ângulos do braço humano usando Visual C# e Kinect	59
Figura 33 – Teste do motor da base do braço	59
Figura 34 – Teste de integração	60

Lista de tabelas

Lista de Gráficos

Lista de abreviaturas e siglas

CSA	Canadian Space Agency
ANFIS	Adaptive Neuro-Fuzzy Inference System
PID	Controlador Proporcional-Integrativo-Derivativo
IoT	Internet of Things
IP	Internet Protocol
RM-ODP	Reference Model of Open Distributed Processing
MPU	Motion Processing Unit
I2C	Inter-Integrated Circuit
PIV	Controlador proporcional-integrativo-velocidade
WiFi	Wireless Fidelity
USB	Universal Serial Bus
HTML	Hypertext Transfer Protocol
TCP-IP	Transmission Control Protocol - Internet Protocol
UDP	User Datagram Protocol
IDE	Integrated Development Environment
PWM	Pulse Width Modulation
JPEG	Joint Photographic Experts Group
CI	Circuito Integrado
MJPEG	Motion JPEG
RAM	Random Access Memory
CPU	Central Processing Unit
ADC	Analog Digital Converter
PC	Personal Computer

FIFO	First In, First Out
J2ME	Java 2 Platform, Micro Edition
J2SE	Java 2 Platform, Standard Edition
J2EE	Java 2 Platform, Enterprise Edition
JVM	Java Virtual Machine
SMC	Simple Managed C
JPL	Jet Propulsion Laboratory
NASA	National Aerospace Agency
VR	Virtual Reality

Sumário

1	INTRODUÇÃO	23
	Introdução	23
1.1	Motivação	23
1.2	Ideias iniciais e alternativas do projeto	24
1.3	Objetivo	25
2	ESPECIFICAÇÕES	27
2.1	Visão Empresarial	27
2.1.1	Monetização do projeto	27
2.1.1.1	Sistema de câmeras estereoscópico	27
2.1.1.2	Braço mecânico	27
2.1.1.3	Projeto como um todo	27
2.2	Visão de Informação	28
2.2.1	Elementos de informação do sistema	28
2.2.1.1	Rotação e aceleração da cabeça do usuário	28
2.2.1.2	Imagens do ambiente remoto	29
2.2.1.3	Ângulos do braço do usuário	29
2.2.2	Manipulação de informações	29
2.2.2.1	Braço robótico	29
2.2.2.2	Câmeras	29
2.2.2.3	Movimentação das Câmeras	29
2.2.3	Fluxo de informações	30
2.3	Visão Computacional	32
2.3.1	Requisitos	32
2.3.1.1	Requisitos funcionais	32
2.3.1.2	Requisitos não funcionais	34
2.4	Visão da Engenharia/Infraestrutura	35
2.4.1	Arquitetura do sistema	35
2.4.1.1	Sistema de Operação do Braço Mecânico	35
2.4.1.1.1	Hardware de captura de dados	36
2.4.1.1.2	Hardware de controle dos motores	36
2.4.1.2	Sistema de Operação do sistema de motores da câmera	36
2.4.1.2.1	Hardware de captura de dados	36
2.4.1.2.2	Hardware de controle dos motores	36
2.4.1.3	Sistema de Imagens Estereoscópicas	36

2.4.1.3.1	Hardware de captura de imagens	36
2.4.1.3.2	Servidor	37
2.4.1.3.3	Hardware de interface gráfica	37
2.5	Visão Tecnológica	37
2.5.1	Microsoft Kinect	38
2.5.2	Particle Photon	39
2.5.3	GY521	40
2.5.4	Raspberry Pi	40
2.5.5	Java	42
2.5.5.1	História	43
2.5.5.2	Características da linguagem	43
2.5.6	Visual C#	43
2.5.6.1	História	44
2.5.6.2	Características da linguagem	44
2.5.7	Protocolo HTTP	45
2.5.8	Servidor Apache HTTP	45
2.5.8.1	Características	46
2.5.9	mjpg-streamer	46
2.5.9.1	Características	46
3	METODOLOGIA	47
3.1	Planejamento	47
3.1.1	Trello	47
3.2	Programas e IDEs	48
3.2.1	Visual Studio 2015	48
3.2.2	Raspberry Pi	49
3.2.3	Particle IDE	49
4	EXECUÇÃO E RESULTADOS OBTIDOS	53
4.0.1	Controle abre-fecha e rotação da garra	53
4.0.2	Modelagem de movimentação do braço	53
4.0.2.1	Modelagem inicial	53
4.0.2.2	Modelagem da movimentação do braço para os motores correspondentes ao ombro, cotovelo e rotação do ombro do braço	54
4.0.2.3	Modelagem da movimentação do braço para os motores correspondentes ao ombro, cotovelo, rotação do ombro e pulso do braço	55
4.0.3	Aquisição de imagens pela camera	55
4.0.4	Montagem do sistema de motores e controle dos motores da câmera	56
4.0.5	Captura de ângulos usando Processing	57
4.0.6	Captura de ângulos usando Visual C#	58

4.0.7	Controle dos servomotores via conexão serial usando Visual C#	58
4.0.8	Controle dos servomotores via conexão UDP usando Visual C# e testes de integração	58
4.1	Resultados	58
5	CONCLUSÃO	61
	REFERÊNCIAS	63

1 Introdução

Hoje, através do uso de tecnologias de telecomunicação, é possível a transmissão de grandes volumes de informações a ambientes remotos sem a necessidade da presença física das partes envolvidas, tornando a interação prática, rápida e fisicamente segura.

Existem casos porém([CSA, 2011](#))([BRUMSON, 2011](#)), em que a interação física a distância com um ambiente remoto é necessária devido a periculosidade ou a dificuldades de acesso ao mesmo, que trariam riscos a um ser humano. Para tal tipo de tarefa, cada vez mais há a substituição de seres humanos por máquinas específicas tais como braços robóticos

Porém, apesar do aumento na segurança para o usuário, muitas das abordagens para a operação de um braço robótico continuam complexas e anti-intuitivas necessitando do uso de artifícios como controles remotos([ICOR, 2016](#)) e joysticks([MITSANTISUK; KATSURA; OHISHI,](#)).

Uma segunda linha de pensamento tenta se aproveitar de semelhanças físicas entre o corpo humano e o braço robótico e faz uso, por exemplo, de sensores ([ALEOTTI; SKOGLUND; DUCKETT,](#)) e marcadores([WANG et al., 2005](#)) presos ao corpo. Entretanto, essas abordagens podem se provar incomodas ou restritivas em relação à movimentação do usuário devido por exemplo, a ocultação de um marcador ou a rigidez de um sensor vestível.([DU et al., 2012](#))

Uma solução para essas restrições, estudada neste trabalho é o uso de tecnologia de rastreamento visual sem marcadores, que permite que o usuário não mais se preocupe com as restrições de movimento impostas, tornando o controle mais natural.([DU et al., 2012](#))

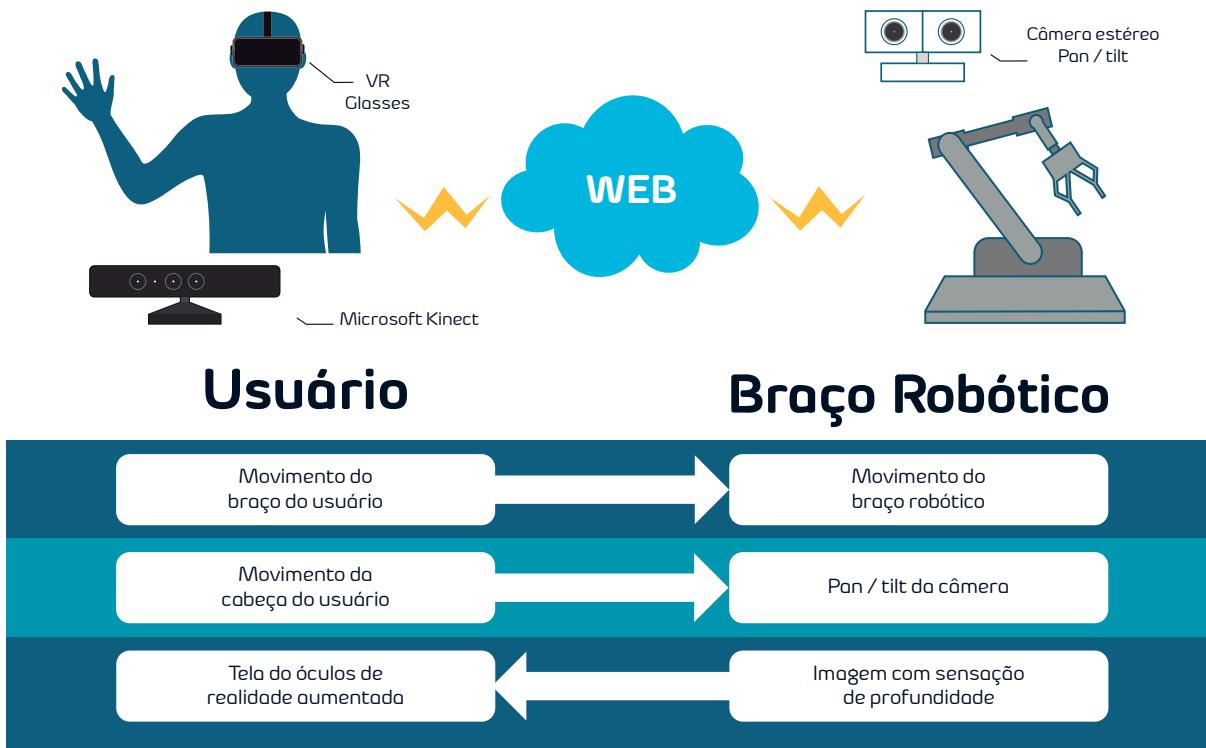
Muitas vezes também, para a operação remota de tais equipamentos, é necessária a transmissão de imagens do ambiente remoto.

Este trabalho estuda o uso de um sistema de imersão visual através do uso de um óculos realidade virtual por rastreamento dos movimentos da cabeça do usuário e um sistema de câmeras estereoscópicas ao projeto, com a finalidade de criar uma experiência imersiva ao usuário.

1.1 Motivação

Este projeto surge da ideia de tentar unir conceitos de engenharia de automação, ênfase da graduação do autor, a conceitos engenharia de computação, área de pesquisa do professor orientador do projeto.

Figura 1 – Diagrama do projeto



Fonte: Autor

Dentro deste contexto multidisciplinar, naturalmente a ideia de um projeto de Internet of Things surge, sendo então discutidas ideias sobre possíveis tema para estudo. Entre as alternativas estudadas, experiências do Jet Propulsion Laboratory da NASA ([JPL](#)) surgem como motivação para este trabalho.

1.2 Ideias iniciais e alternativas do projeto

Inicialmente, a pesquisa inicial tratou os seguintes temas:

- Geração de trajetória para braço robótico usando Microsoft Kinect;
- Captura de movimentos usando giroscópio;
- Algoritmos anti-drifting de giroscópio;
- Captura de imagens estereoscópicas em tempo real;
- Arduino;
- Raspberry Pi;
- Particle Photon;
- Internet of Things;

As ideias iniciais continham premissas que envolviam o uso de redes neurais para o controle do braço mecânico, utilizando um controlador ANFIS-PID para o controle dos motores, chegando a ser testado.

Posteriormente, em virtude de limitações de memória do hardware e a complexidade adicionada ao projeto, os algoritmos de redes neurais foram excluídos do escopo do projeto. O formato final é tratado na seção seguinte([seção 1.3](#)).

1.3 Objetivo

Este estudo propõe um sistema de imersão e interação do usuário em um ambiente remoto através da operação de um braço mecânico e do uso de um sistema de imersão visual.

A face de interação do projeto é representada pela operação de um braço robótico à distância, realizada através do uso do sensor de captura de movimento Microsoft Kinect e de um braço robótico conectado a uma rede WiFi.

O segundo objetivo do projeto trata da imersão visual do usuário no ambiente remoto em que ocorre a interação. Tal imersão será realizada através de um sistema de câmeras estereoscópicas com motores de rotação pan/tilt, uso de óculos de realidade virtual para a recepção das imagens das câmeras e rastreamento de movimentos da cabeça do usuário para o controle de movimentação dos motores.

Visando uma maior adaptabilidade de suas partes a outras aplicações, o projeto será encarado de forma modular, de forma que as partes componentes descritas acima poderão ser utilizadas independentemente.

Além disso, o rigor metodológico é uma das preocupações centrais deste trabalho, o que inclui um refinamento na maneira de definir os requisitos do sistema, representado pelo RM-ODP.

2 Especificações

Esta seção baseia-se nos 5 pontos fundamentais da ODP (Open Distributed Processing) para especificar o projeto a ser desenvolvido. Assim, divide-se este capítulo em: Visão Empresarial, de Informação, Computacional, de Engenharia/Infraestrutura e de Tecnologia.

2.1 Visão Empresarial

2.1.1 Monetização do projeto

O projeto desenvolvido tem o intuito de possibilitar a interação imersiva com um ambiente remoto através do uso de câmeras estereoscópicas e da operação de um braço robótico. Analisando o projeto, encontramos usos distintos para suas partes componentes em separado, assim como usos envolvendo as duas partes.

2.1.1.1 Sistema de câmeras estereoscópico

- Câmeras de Vigilância de Ambientes: é possível aplicá-las ao utilizar mais de um conjunto de câmeras, uma vez que o usuário tem a possibilidade de assinalar um endereço independente para cada um deles.
- Câmeras de Entretenimento: ao possibilitar o acompanhamento visual baseado na movimentação da cabeça do usuário, é possível dar acesso a áreas restritas de locais como exposições, zoológicos, aquários e áreas sensíveis a concentração de pessoas através das câmeras instaladas, permitindo uma experiência exclusiva em tais locais.

2.1.1.2 Braço mecânico

- Programação de braços robóticos industriais : Caso seja desenvolvido um software para o armazenamento dos dados de movimentação do usuário, é possível a utilização do projeto para uma rápida automatização de uma tarefa repetitiva.

2.1.1.3 Projeto como um todo

- Desarmamento de Explosivos: pode ser utilizado em operações de reconhecimento em locais remotos e no desarmamento de explosivos, proporcionando uma maior segurança aos operadores.
- Operações Médicas à Distância: Este tipo de abordagem pode oferecer novas alternativas para operações à distância mais eficazes e flexíveis, dando a locais com escassez

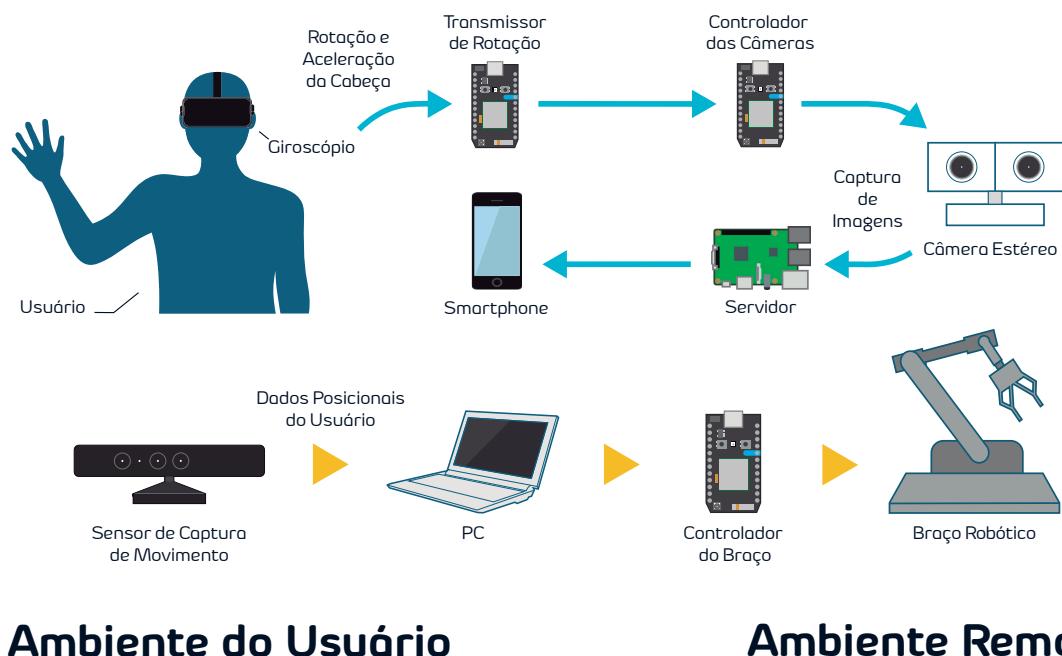
de recursos acesso a serviços médicos. Para tal aplicação no entanto, são necessárias melhorias na precisão do hardware/software.

- Jogos de Realidade Aumentada: o projeto pode ser viabilizado neste campo, dando uma experiência imersiva ainda mais enriquecedor ao usuário não só no ambiente virtual, mas também no ambiente real mostrando um grande potencial na criação de novas dinâmicas de jogos e entretenimento.

2.2 Visão de Informação

2.2.1 Elementos de informação do sistema

Figura 2 – Elementos de informação do sistema



Fonte: Autor

Para facilitade de compreensão, esta subseção se divide em três partes:

2.2.1.1 Rotação e aceleração da cabeça do usuário

Os movimentos verticais e horizontais da cabeça do usuário são coletados e convertidos em valores numéricos de aceleração e ângulo através do giroscópio GY-521 acoplado na parte traseira do óculos de realidade virtual (Figura 2).

2.2.1.2 Imagens do ambiente remoto

As imagens são coletadas por duas câmeras e enviadas ao servidor(Raspberry Pi)(Figura 2).

2.2.1.3 Ângulos do braço do usuário

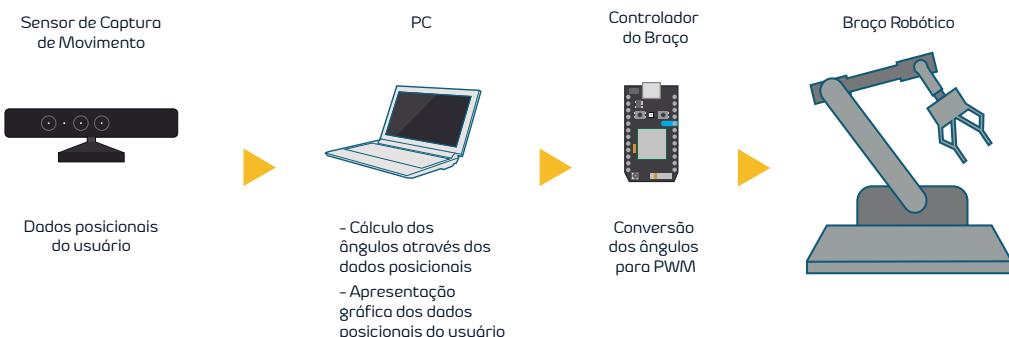
Os movimentos são coletadas através de um sensor de captura de movimentos e enviadas a um computador, que serve como um segundo servidor(Figura 2).

2.2.2 Manipulação de informações

2.2.2.1 Braço robótico

As informações de posição espacial do usuário são enviadas pelo sensor de movimentos a um computador, que por sua vez converte os dados em ângulos e retransmite ao controlador, responsável por fazer a ponte entre o software e os motores do braço mecânico(Figura 3).

Figura 3 – Manipulação de dados pelo sistema do braço robótico



Fonte: Autor

2.2.2.2 Câmeras

As imagens do ambiente remoto são capturadas por duas câmeras paralelas e enviadas a um servidor(Figura 4).

2.2.2.3 Movimentação das Câmeras

Os dados de movimentação da cabeça do usuário são enviados para um Transmissor de ângulo de rotação(Figura 5).

Figura 4 – Manipulação de dados pelo sistema de câmeras estereoscópicas



Fonte: Autor

Figura 5 – Manipulação de dados pelo sistema de motores da câmera



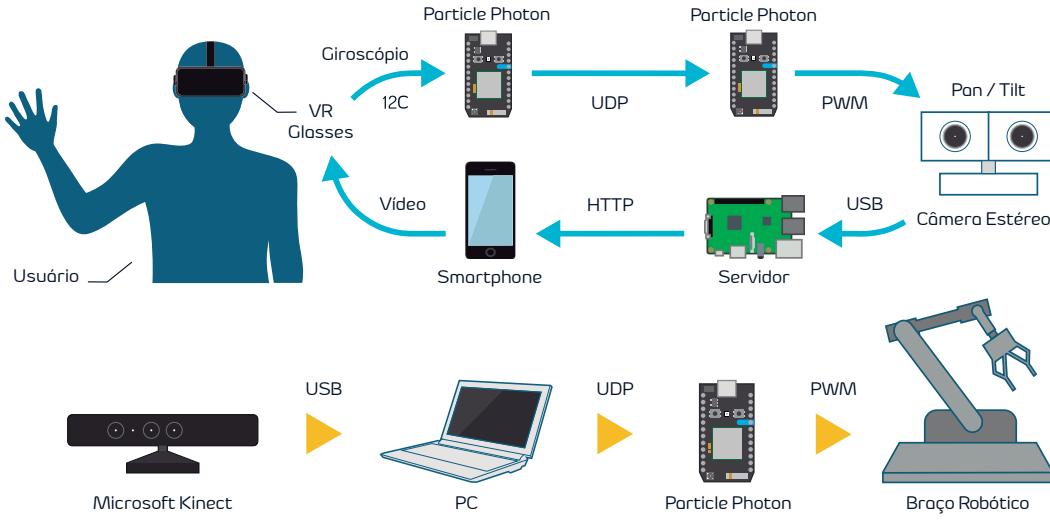
Fonte: Autor

2.2.3 Fluxo de informações

No projeto, todo o fluxo de informações entre os nós da arquitetura é unidirecional. Na Figura 6, em cada flecha de comunicação, é mostrada de forma genérica a tecnologia utilizada entre os nós para troca de informações. As tecnologias para intercomunicação são as seguintes:

- PWM - Link Cabeado: Comunicação entre os motores e o receptor de ângulos da câmera.
- PWM - Comunicação entre os motores e o receptor de ângulos do braço.
- Serial USB - Link Cabeado: Comunicação entre as câmeras e o servidor.
- Serial USB - Link Cabeado: Comunicação entre sensor de captura de movimento e o computador(servidor)
- Serial I2C - Link Cabeado: Comunicação entre giroscópio GY521 e Particle Photon.

Figura 6 – Tecnologias Utilizadas para Conexão entre os Nós



Ambiente do Usuário

Ambiente Remoto

Fonte: Autor

- HTTP TCP-IP- Link Internet: Uso do protocolo HTTP e TCP-IP para comunicação entre dispositivo móvel e servidor.
- Wireless UDP- Link Wireless: Uso do protocolo UDP para comunicação entre transmissor de dados de rotação da cabeça do usuário e o controlador dos motores da câmera.
- Wireless UDP- Link Wireless: Uso do protocolo UDP para comunicação entre o computador e o controlador do braço robótico.

As mensagens trocadas entre cada nó são detalhadas a seguir:

- Transmissor e Receptor da câmera: O transmissor de dados de rotação da cabeça do usuário envia ao o controlador dos motores da câmera mensagens com o seguinte formato:
ângulos : “rotação horizontal” : dados[0], “rotação vertical” : dados[1]
- Giroscópio e Transmissor de ângulos de rotação: O giroscópio envia ao transmissor de dados de rotação da cabeça do usuário quaternion com o seguinte formato: ângulos : “yaw” : dados[0], “pitch” : dados[1], “row” : dados[2], “rotation” : dados[3]

- Transmissor e Receptor da câmera: O transmissor de dados de rotação da cabeça do usuário envia ao controlador dos motores da câmera mensagens com o seguinte formato:

ângulos : “motor da base” : dados[0], “motor do ombro” : dados[1], “motor do cotovelo” : dados[2], “motor do pulso” : dados[3], “motor de rotação do pulso” : dados[4], “motor da garra” : dados[5]

- Receptor(ambos) e Motores: O receptor converte os dados recebidos em comandos PWM para os servo motores.
- Câmeras e Servidor: As câmeras enviam imagens no formato JPEG para o computador em que o servidor se encontra(Raspberry Pi), que são coletadas pelo servidor.
- Smartphone e Servidor: A aplicação utiliza o protocolo HTTP para realizar requisições ao servidor Apache pela internet.

2.3 Visão Computacional

2.3.1 Requisitos

2.3.1.1 Requisitos funcionais

Nesta seção, apresentam-se os requisitos funcionais e não funcionais do sistema proposto.

Identificação	RFB01
Nome	Captura de movimentos (braço)
Descrição	O sistema deve suportar a captura da posição espacial das juntas do corpo do usuário
Explicação	Os dados serão usados para o cálculo dos ângulos de movimentação do braço robótico

Identificação	RFB02
Nome	Cálculo de ângulos (braço)
Descrição	O sistema deve receber a posição espacial das juntas do corpo do usuário e através de trigonometria, calcular os ângulos de cada junta do braço do usuário
Explicação	Os ângulos calculados serão transmitidos para o controlador dos motores do braço robótico

Identificação	RFB03
Nome	Visualização de ângulos
Descrição	O sistema deve exibir o "esqueleto" do usuário e mostrar os ângulos em cada junta do braço
Explicação	O sistema deve possuir uma interface gráfica que permita a verificação de captura dos dados e seu correto processamento em tempo real.

Identificação	RFC01
Nome	Recepção de imagens(câmera)
Descrição	O sistema deve receber as imagens das duas câmeras do sistema e disponibilizá-las ao servidor
Explicação	As imagens serão juntadas pelo servidor em uma imagem estereoscópica, que será transmitida ao usuário

Identificação	RFC01
Nome	Conversão de imagem
Descrição	O sistema deve receber a imagem das câmeras e formatá-las de modo apropriado à visão estereoscópica
Explicação	As duas imagens serão disponibilizadas lado a lado, de forma que sejam observadas como uma única imagem com profundidade quando o usuário vestir os óculos de realidade virtual.

Identificação	RFM01
Nome	Cálculo de ângulos(motores da câmera)
Descrição	O sistema deve receber os dados de rotação e aceleração do giroscópio e calcular os ângulos em cada eixo com as devidas correções
Explicação	Os ângulos calculados serão transmitidos para o controlador dos motores do braço robótico

Identificação	RFM02
Nome	Cálculo de ângulos (motores da câmera)
Descrição	O sistema deve receber a posição espacial das juntas do corpo do usuário e através de trigonometria, calcular os ângulos de cada junta do braço do usuário
Explicação	Os ângulos calculados serão transmitidos para o controlador dos motores do braço robótico

2.3.1.2 Requisitos não funcionais

Identificação	RNFB01
Nome	Taxa de envio de dados (braço)
Descrição	O hardware de comunicação deve otimizar o número de requisições enviadas ao hardware de controle do braço.
Explicação	A taxa de envio de dados deve ser suficiente para que o intervalo entre envios não seja perceptível, sem que sobre gere o hardware de controle.

Identificação	RNFB01
Nome	Taxa de envio de dados (Câmeras)
Descrição	O hardware de captura de imagens deve otimizar o número de imagens enviadas ao servidor.
Explicação	A taxa de envio de dados deve ser suficiente para que o intervalo entre frames não seja perceptível, sem que sobrecarregue o servidor.

Identificação	RNFM01
Nome	Taxa de envio de dados (motores da câmera)
Descrição	O hardware de comunicação deve otimizar o número de requisições enviadas ao hardware de controle dos motores da câmera.
Explicação	A taxa de envio de dados deve ser suficiente para que o intervalo entre envios não seja perceptível, sem que sobrecarregue o hardware de controle.

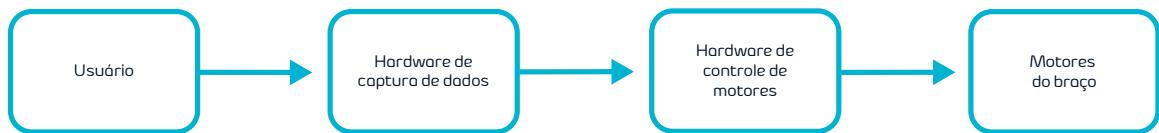
2.4 Visão da Engenharia/Infraestrutura

Esta seção apresenta a arquitetura definida para a implementação do sistema.

2.4.1 Arquitetura do sistema

2.4.1.1 Sistema de Operação do Braço Mecânico

Figura 7 – Arquitetura do sistema de operação do braço robótico



Fonte: Autor

2.4.1.1.1 Hardware de captura de dados

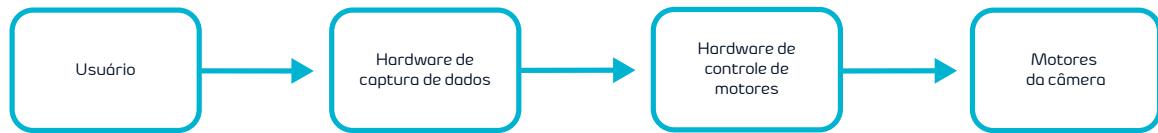
É o hardware que realiza a interface com o usuário. Trata-se da parte responsável por coletar os dados relevantes de movimentação do braço do usuário, convertê-los para ângulos e enviá-los ao controlador do motor via protocolo UDP.

2.4.1.1.2 Hardware de controle dos motores

É o hardware que realiza a interface com os motores, convertendo os dados recebidos em comandos PWM usados pelos motores.

2.4.1.2 Sistema de Operação do sistema de motores da câmera

Figura 8 – Arquitetura do sistema de motores da câmera



Fonte: Autor

2.4.1.2.1 Hardware de captura de dados

Analogamente ao hardware do braço, é o hardware que realiza a interface com o usuário.

Trata-se da parte responsável por coletar os dados relevantes de movimentação do braço do usuário, convertê-los para ângulos e enviá-los ao controlador do motor via protocolo UDP.

2.4.1.2.2 Hardware de controle dos motores

Novamente, é o hardware que realiza a interface com os motores, convertendo os dados recebidos em comandos PWM usados pelos motores.

2.4.1.3 Sistema de Imagens Estereoscópicas

2.4.1.3.1 Hardware de captura de imagens

Realiza a captura de imagens do ambiente remoto e as disponibiliza para o servidor.

Figura 9 – Arquitetura do sistema de captura de imagens estereoscópicas



Fonte: Autor

2.4.1.3.2 Servidor

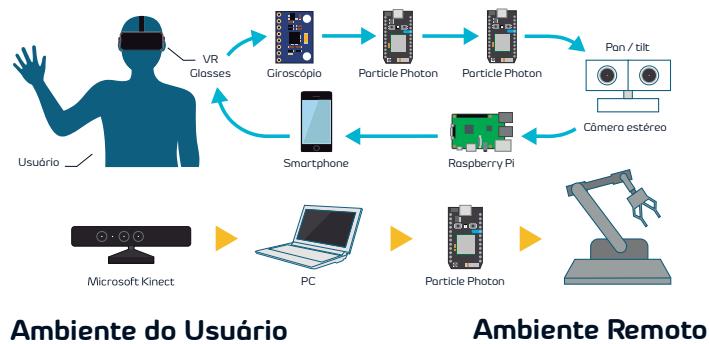
Recebe as imagens e as reformata para que sejam vistas em um óculos de realidade virtual genérico como uma imagem estereoscópica.

2.4.1.3.3 Hardware de interface gráfica

Realiza a requisição de imagens ao servidor. Além disso, é responsável pela interface com o usuário, isto é, permite a visualização do ambiente remoto.

2.5 Visão Tecnológica

Figura 10 – Visão tecnológica de Hardware do sistema



Fonte: Autor

Neste capítulo, são apresentadas as tecnologias utilizadas para cada camada do projeto, a partir da arquitetura definida no seção anterior . Posteriormente, são fornecidas informações básicas sobre cada tecnologia. As principais tecnologias utilizadas foram:

Microsoft Kinect, Raspberry Pi, Particle Photon, GY521, Visual C#, Apache HTTP Server, mjpg-streamer.

2.5.1 Microsoft Kinect

Figura 11 – Logotipo Kinect



Fonte: Microsoft

Figura 12 – Kinect 2.0



Fonte: Microsoft

Originalmente chamado Projeto Natal durante seu desenvolvimento, Kinect é uma linha de sensores de captura de movimento desenvolvida pela Microsoft em parceria com a Invensense para seus consoles Xbox 360 e Xbox one.

Baseado em um periférico add-on estilo webcam, ele permite que os usuários controlem e interajam com seus consoles/computadores sem a necessidade de um controle remoto, através de uma interface natural do usuário usando gestos e comando por voz. A primeira geração do Kinect foi introduzida em novembro de 2010, com uma tentativa de expandir o público de Xbox 360 além do seu público gamer usual.

Uma versão para Windows foi lançada em 1 de fevereiro de 2012, e em 16 de junho de 2011 a Microsoft lançou o kit de desenvolvimento de software do Kinect para Windows 7. Este SDK permitiu a desenvolvedores escreverem apps Kinect em C++/CLI, C#, ou

Visual Basic .NET. Atualmente, o Kinect encontra-se em sua segunda versão, com sua SDK na versão 2.0([MICROSOFT, a](#)).

2.5.2 Particle Photon

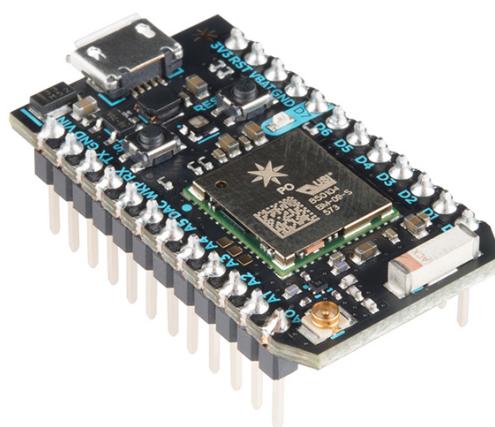
Figura 13 – Logotipo Particle



Fonte: Particle

A Particle surgiu como uma campanha no Kickstarter em 2013 com a visão de desenvolvimento de Internet of Things simples e acessível e atualmente suas ferramentas são utilizados por 70 mil engenheiros em mais de 170 países e por várias companhias Fortune 500 para desenvolver e gerenciar diversos novos produtos IoT. Seu portfólio de produtos inclui a Particle Cloud, uma infraestrutura de nuvem própria para os dispositivos da empresa, um cartão SIM e plano de datas próprios, os microcontroladores conectados via nuvem Electron e Photon e software exclusivo para desenvolvimento. A Particle foi listada em 2015 como uma das Fast Company's Most Innovative Companies e é listada em vários relatórios da Gartner em soluções IoT.

Figura 14 – Particle Photon



Fonte: Particle

A Particle Photon, utilizada neste projeto é uma placa de prototipagem de hardware e software voltada a IoT baseada em um microcontrolador STM32 ARM Cortex M3 e um CI WiFi Broadcom BCM43362 totalmente desenvolvida pela Particle. A placa foi escolhida pela fácil programação, fator de forma e peso reduzidos e pela conectividade WiFi integrada([PARTICLE](#),).

2.5.3 GY521

A placa GY521 é uma placa destinada a tornar a prototipação usando o CI MPU6050 mais fácil e prática ao integrá-lo em uma placa com saídas through hole.

O CI MPU6050 é um dispositivo de rastreamento de movimento que contém um giroscópio de 3 eixos, um acelerômetro de 3 eixos e um Processador digital de movimento(DMP) em um encapsulamento de 4x4x0.9mm.

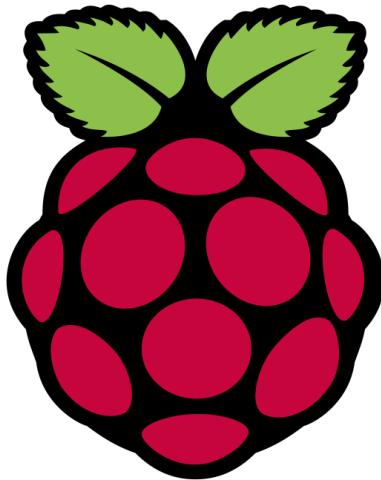
A MPU-6050 possui três conversores analógico-digitais de 16 bits (ADCs) para digitalizar as saídas do giroscópio e três 16-bit ADCs para digitalizar as saídas do acelerômetro. Para rastreamento de precisão de movimentos rápidos e lentos, o giroscópio apresenta ranges programáveis pelo usuário de ± 250 , ± 500 , ± 1000 , e $\pm 2000^{\circ}/sec$ (dps) e o acelerômetro, ranges de $\pm 2g$, $\pm 4g$, $\pm 8g$, e $\pm 16g$. Um buffer on-chip FIFO de 1024 Bytes ajuda na redução de consumo de energia do sistema permitindo ao processador de sistema ler os dados do sensor em bursts e assim, entrar em um modo de baixa potência enquanto o MPU coleta mais dados. Comunicação com todos os registros de dispositivos são executadas utilizando ou I2C, ou 400kHz. Recursos adicionais incluem um sensor de temperatura acoplado e um oscilador on-chip com variação $\pm 1\%$ na faixa de temperatura de operação. A peça possui uma robusta tolerância a choques de 10.000g e possui filtros passa-baixa programáveis para o giroscópio, o acelerômetro e o sensor de temperatura([NADA](#), a).

2.5.4 Raspberry Pi

A ideia para um computador pequeno e barato para crianças surgiu em 2006, quando Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft, do laboratório de computação da Universidade de Cambridge, ficaram preocupados com o declínio do nível dos estudantes do ensino médio que pleiteavam uma vaga para o curso de ciência da computação. Nos anos 90, os estudantes entrevistados para o curso tinham uma vasta experiência como programadores hobbyistas, no entanto nos anos 2000 a situação mudou bastante os candidatos em sua maioria tinham algum conhecimento em web design.

O jeito como as crianças inglesas lidam com a tecnologia mudou. Alguns problemas foram encontrados: um grande número de aulas utilizando Word e Excel, ou escrevendo páginas web; o fim do crescimento da era ponto-com; e o crescimento dos PC's e consoles de video-game, que substituíram as máquinas que as pessoas das gerações anteriores

Figura 15 – Logotipo Raspberry Pi



Fonte: Raspberry Pi Foundation

Figura 16 – Raspberry Pi Modelo B versão 3



Fonte: Raspberry Pi Foundation

aprenderem a programar.

Não há muito em que um pequeno grupo possa fazer para solucionar problemas como um currículo inadequado ou o fim de uma bolha financeira. Mas o grupo de Cambridge achou que podia fazer algo para mudar a situação em que os computadores se tornaram custosos e complexos e em que a programação neles teve que ser proibida pelos pais e assim pensaram em uma plataforma, que assim como os computadores pessoais antigos, podiam inicializar em um ambiente de programação. Então de 2006 a 2008, este grupo desenvolveu o que agora se tornou o Raspberry pi.

Em 2008, os processadores desenvolvidos para telefones celulares se tornaram mais acessíveis, e com capacidade de processamento suficiente para prover multimídia, um recurso que poderia deixar a placa atraente para crianças, que não se interessariam por um dispositivo puramente voltado para programação.

Foi então que foi criada a Fundação Raspberry pi, para transformar o projeto em realidade. Três anos depois o Raspberry pi modelo B entrou em produção em massa, e em dois anos vendeu mais de dois milhões de unidades.(...)

O objetivo do Raspberry é difundir o uso de computadores de baixo custo, que possam ser utilizados para programação. Sendo uma tentativa de quebrar o paradigma que para ter acesso a internet é necessário comprar um computador de alto custo. Outra meta é a de que o uso dos computadores pessoais seja difundido entre as crianças. ([FOUNDATION, a](#)).

2.5.5 Java

O desenvolvimento em Processing é realizado utilizando-se Java, linguagem de programação orientada a objetos. Figura 31 – Logo da linguagem Java

Figura 17 – Logo da linguagem Java



Fonte: Oracle

2.5.5.1 História

Java foi iniciado em junho de 2001, em um projeto liderado por James Gosling, Mike Sheridan e Patrick Naughton, sendo originalmente projetado para televisão interativa.

Inicialmente, a linguagem era chamada Oak, referência ao carvalho (oak) plantado no exterior do escritório de Gosling. Mais tarde, a linguagem foi chamada de Green e, por fim, Java.

A primeira implementação pública do Java (Java 1.0) foi lançada em 1995, pela Sun Microsystems. Nesta época, os principais navegadores incorporaram a execução de Java applets nas páginas web. Posteriormente, com o lançamento da versão Java 2, aumentou o número de plataformas e configurações compatíveis. A versão destinada à Desktops foi denominada J2SE (Java Standard Edition), enquanto as aplicações de empresas eram desenvolvidas utilizando-se a tecnologia J2EE (Java Enterprise Edition). Por sua vez, a versão J2ME (Java Mobile Edition) oferecia recursos especificamente para aplicações de celulares. Posteriormente, por motivos de marketing, as versões foram renomeadas Java SE, Java EE e Java ME.

Atualmente, o Java SE encontra-se na versão 8.0, e é considerada ainda como uma das mais utilizadas linguagens de programação. ([ORACLE](#),)

2.5.5.2 Características da linguagem

O código Java é compilado em um bytecode, que é executado na máquina virtual java (JVM - Java Virtual Machine), independentemente da arquitetura da máquina. Possui gestão automática de memória, com uma implementação padrão de “Garbage Collector” que monitora as referências aos objetos em uma aplicação. Graças à grande popularidade da linguagem, existem inúmeras bibliotecas e frameworks disponíveis atualmente para o desenvolvimento de aplicações em Java. Dentre as funcionalidades oferecidas pelas bibliotecas mais utilizadas, destacam-se: IO (Input/Output), etworking (comunicação em rede), Concurrency (programação paralela), segurança, interfaces gráficas, etc.

2.5.6 Visual C#

C# é uma linguagem de programação de propósito geral orientada a objetos criada para o desenvolvimento de uma variedade de aplicações executadas sobre o .NET Framework.

Seu time de desenvolvimento é conduzido por Anders Hejlsberg e sua versão mais recente é C# 6.0, lançada em 2015.

Já o Visual C#, uma implementação da linguagem C# pela Microsoft, é suportado por Visual Studio que possui um editor de código completo, compilador, modelos de

projetos, designers, assistentes de código, um depurador poderoso e de fácil manuseio e entre outras ferramentas.

A biblioteca de classes do .NET Framework fornece acesso a vários serviços do sistema operacional.

2.5.6.1 História

Durante o desenvolvimento de .NET Framework, a biblioteca de classes foi originalmente escrita utilizando um sistema compilador de códigos gerenciados chamado Simple Managed C (SMC). Em janeiro de 1999, Anders Hejlsberg forma um time com o objetivo de construir uma nova linguagem que inicialmente foi nomeada Cool e posteriormente alterado para "C-like Object Oriented Language". Apesar de ter sido cogitado manter o nome "Cool" como nome final da linguagem, por questões de marca registrada esta ideia foi abandonada pela Microsoft.

Em 2000 quando o projeto .NET foi anunciado publicamente, a linguagem já havia sido renomeada para C# e as bibliotecas de classes e a ASP.NET runtime já haviam sido transferida para esta linguagem. O principal designer e arquiteto da C# foi Anders Hejlsberg da Microsoft, que já havia participado no desenvolvimento de Turbo Pascal, Embarcadero Delphi (antigos CodeGear Delphi, Inprise Delphi e Borland Delphi), e Visual J++.

(MICROSOFT, b)

2.5.6.2 Características da linguagem

C# foi projetado para se adequar a aplicações tanto em sistemas hospedados quanto embarcados, que vão desde grandes aplicações com sistemas operacionais sofisticadas até as menores com funções específicas. Devido a estas características, tanto a linguagem como suas implementações devem fornecer suporte para princípios de engenharia de software como checagem strong type, checagem de array bounds, detecção de tentativas de utilização de variáveis não inicializadas e coletor de lixo automático. Robustez e durabilidade de software e a produtividade do programador são fatores importantes para a linguagem.

Portabilidade é também uma característica muito importante para códigos-fonte e para programadores, especialmente para aqueles que já estão familiarizados com C and C++.

Finalmente, embora as aplicações C# terem sido planejadas para serem econômicas considerando a memória e requisitos de poder de processamento, a linguagem não se destina a competir diretamente em desempenho e tamanho com C ou Assembly.

2.5.7 Protocolo HTTP

HTTP (HyperText Transfer Protocol) é um protocolo da camada de aplicação para comunicação via internet. Trata-se de um protocolo do tipo requisição-resposta, baseado no modelo de computação cliente-servidor, no qual o cliente envia requisições ao servidor que, por sua vez, retorna uma resposta. Por exemplo, um navegador web é um cliente, enquanto um outro computador hospedando uma página web é o servidor.

No contexto do modelo de camadas de redes, HTTP é um protocolo da camada de aplicação e é comumente utilizado em conjunto do TCP (Transmission Control Protocol), protocolo da camada de transporte. Dentre os métodos de requisições mais comuns, destacam-se:

GET

Requisita uma representação de um determinado recurso (uma página da internet, por exemplo). Requisições utilizando GET devem apenas recuperar dados e não ter outro efeito.

POST

Requisita que o servidor web aceite e salve os dados enviados no corpo (body) da mensagem enviada. É frequentemente utilizado ao fazer uploads de arquivos ou submeter formulários. Neste projeto, requisições do tipo GET são utilizadas para transmitir as imagens usando o servidor.

2.5.8 Servidor Apache HTTP

Figura 18 – Logo Apache HTTP Server



Fonte: Apache Software Foundation

O Projeto Servidor Apache HTTP é um esforço de desenvolvimento de software colaborativo destinado a criar implementação de código fonte de servidor HTTP (Web) robusto, comercial, característico e disponível gratuitamente. O projeto é administrado em conjunto por um grupo de voluntários de diversos países, usando a internet e a web para comunicar, planejar e desenvolver o servidor e sua documentação relacionada. O Apache HTTP Server é um projeto da Apache Software Foundation.

2.5.8.1 Características

Lançado em 1995, o projeto Apache HTTP Server ("httpd") da fundação The Apache Software Foundation tem sido o servidor web mais popular desde abril de 1996, tendo sido celebrado o 20º aniversário do projeto em fevereiro de 2015.([FOUNDATION](#), b)

2.5.9 mjpg-streamer

mjpg-streamer é uma aplicação de linhas de comando criada originalmente por Tom Stöveken para linux que copia frames JPEG de uma ou mais fontes para múltiplos output plugins. Pode ser usada para transmitir arquivos JPEG através de uma webcam conectada a uma rede baseada em IP para vários tipos de software capazes de receber transmissões MJPG ao mesmo tempo.

2.5.9.1 Características

Originalmente escrito para aparelhos embarcados com memórias RAM e CPU limitadas, uvc_streamer(programa predecessor ao mjpg-streamer) foi criado devido ao fato de câmeras compatíveis com Linux-UVC produzirem diretamente dados em JPEG, o que permitia uma transmissão M-JPEG rápida e de com performance considerável. Hoje, mjpg-streamer tem suporte a uma variedade de dispositivos de entrada diferentes.([STÖVEKEN](#),)

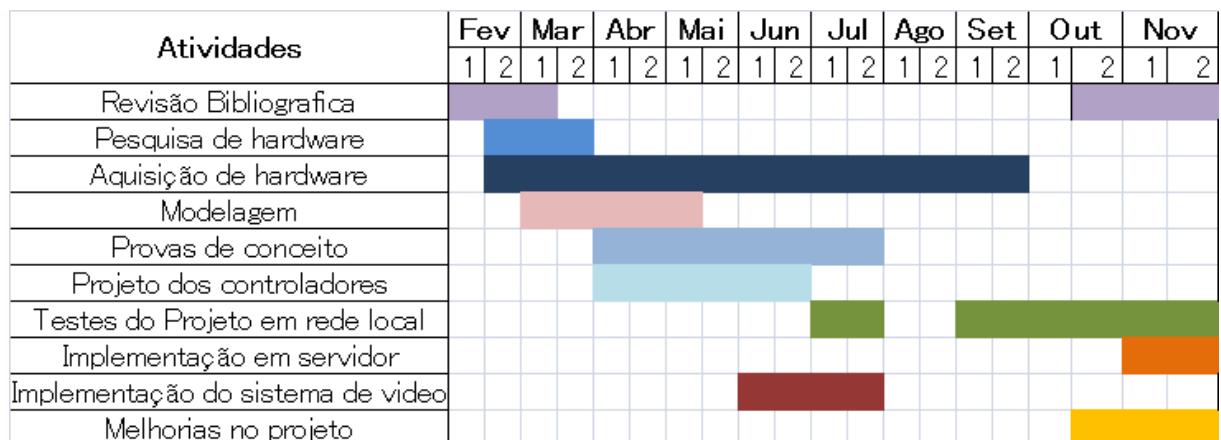
3 Metodologia

Este capítulo apresenta o método de trabalho adotado para a execução das atividades do projeto e divide-se em três partes: organização de tarefas, escolha de hardware e software utilizados.

3.1 Planejamento

Inicialmente, foi elaborado um diagrama de Gantt do sistema contendo as principais etapas do projeto (Figura 19) a partir do qual foram identificadas as tarefas a serem realizadas durante o projeto, que foram posteriormente transferidas para o Trello.

Figura 19 – Diagrama de Gantt do sistema



Fonte: Autor

3.1.1 Trello

Trello é uma ferramenta de organização de projetos baseada nos quadros Kanban da metodologia ágil. Extremamente versátil, permite uma análise rápida da situação do projeto, das etapas cumpridas e do cronograma geral do projeto. No Trello, é possível criar “boards” que agrupam listas (“lists”) de tarefas que devem ser feitas. Cada tarefa é então representada por um “card”, criado dentro de uma “list”. As “lists” criadas dentro da “board” do projeto são:

Backlog : Tarefas que ainda precisam ser analisadas e aprovadas para serem colocadas na fase de atividades.

Sprint : Tarefas a serem realizadas após a conclusão das tarefas sendo trabalhadas atualmente.

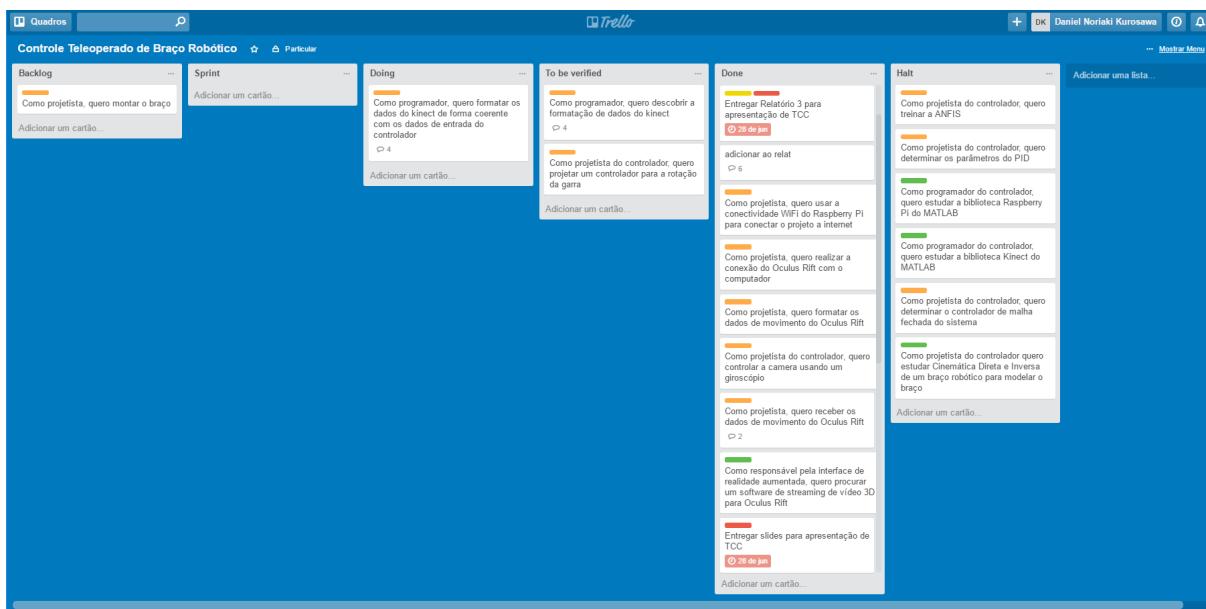
Doing : Tarefas em execução.

To be verified : Tarefas executadas a serem revisadas.

Done : Tarefas terminadas.

Halt : Tarefas suspensas/canceladas.

Figura 20 – Kanban do projeto usando Trello



Fonte: Autor

3.2 Programas e IDEs

Esta seção apresenta os principais programas e IDEs (Integrated Development Environment) utilizados durante o desenvolvimento de cada componente do sistema proposto.

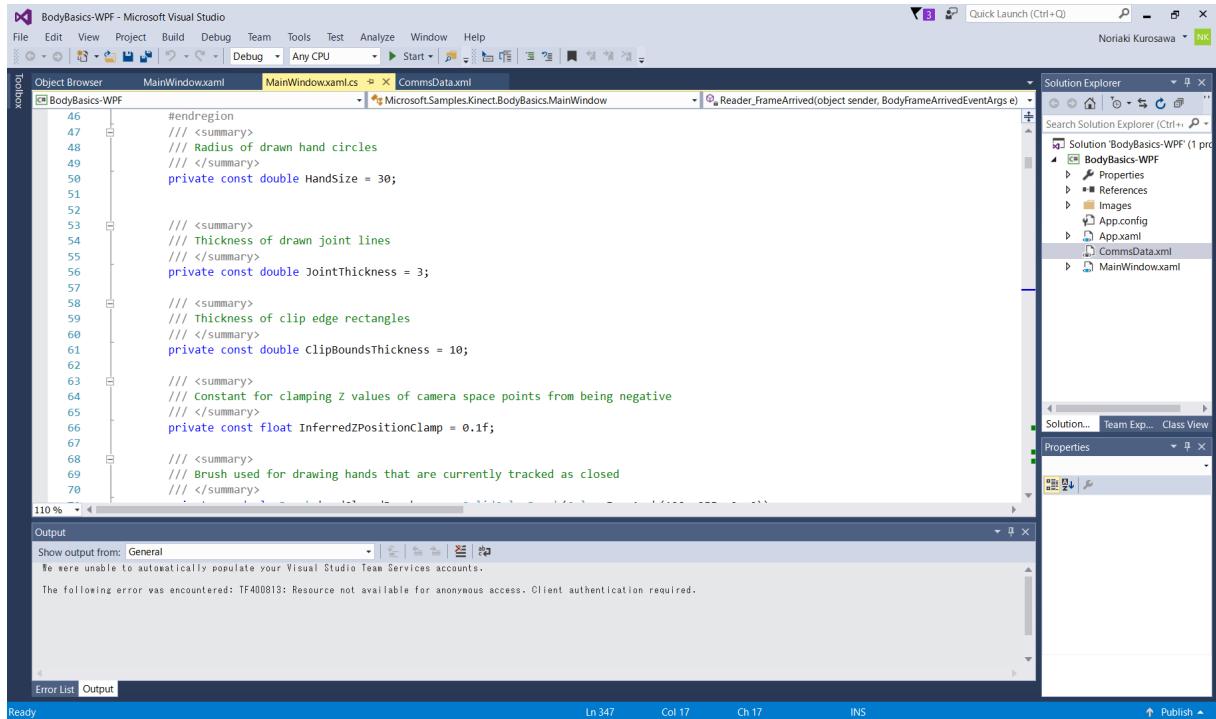
3.2.1 Visual Studio 2015

A aplicação para Kinect foi feita usando-se o programa Visual Studio 2015.

O Visual Studio é a IDE usada pelo Kinect SDK da Microsoft para o desenvolvimento de aplicações Kinect, e permite elaborar o aplicativo e facilmente verificar os seus

erros. É também a Possui inúmeros exemplos de código de aplicações em C#, C++ e Visual Basic que também auxiliam no desenvolvimento de uma aplicação.

Figura 21 – Visual Studio 2015 IDE



Fonte: Autor

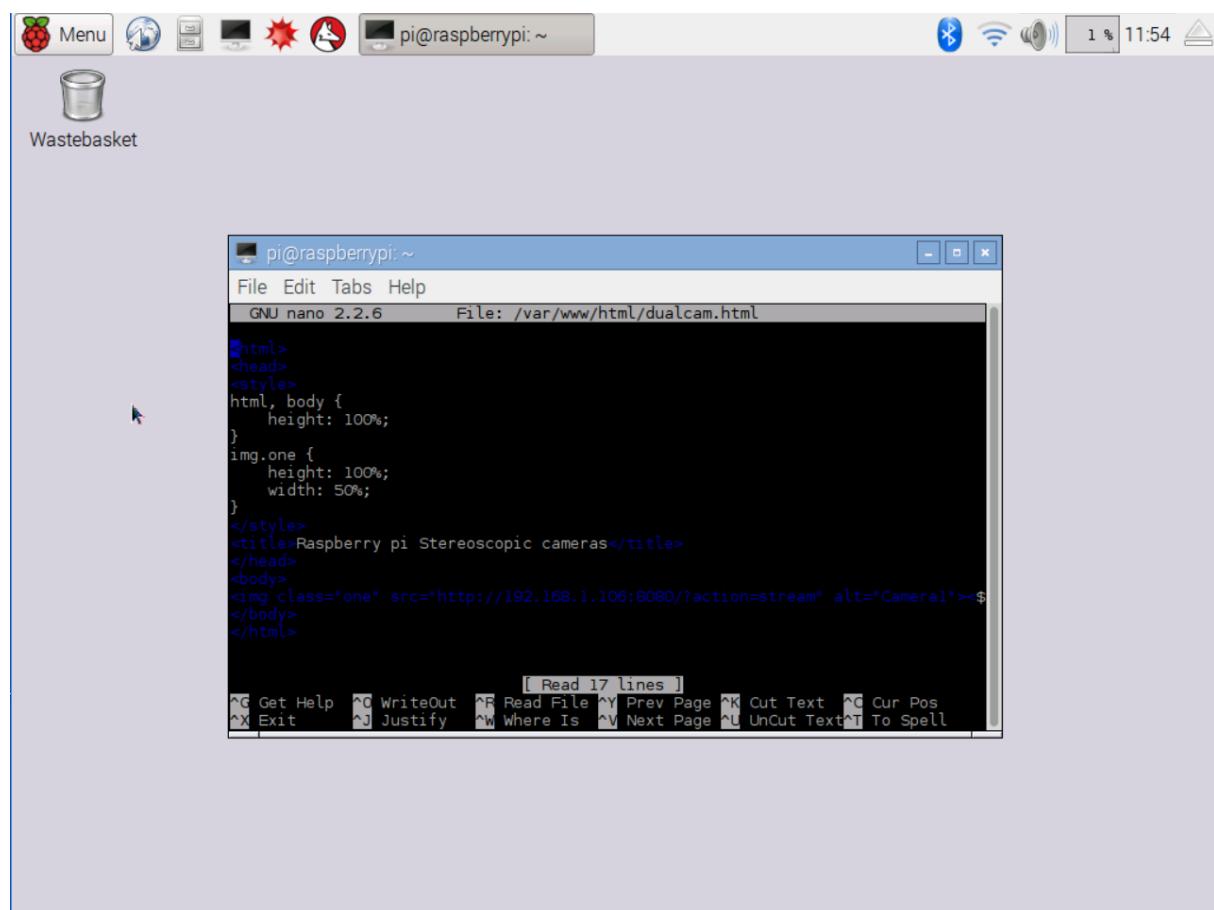
3.2.2 Raspberry Pi

A configuração do servidor HTTP Apache foi escrita utilizando-se o editor de texto GNU NANO, que já vem pré-instalado no Raspberry Pi. A escolha se deu pela simplicidade da estrutura do arquivo de configuração, que não demandava grandes funcionalidades.

3.2.3 Particle IDE

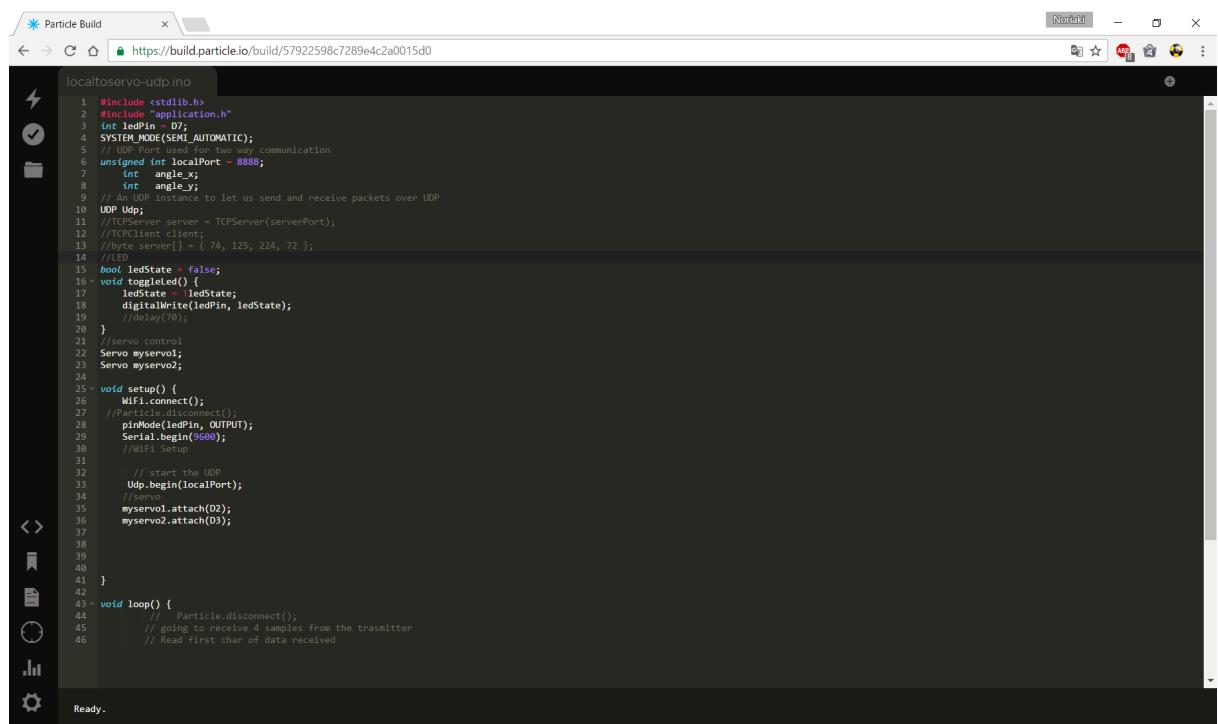
A Programação das placas Photon se deu pelo uso da Particle IDE, ambiente próprio para a programação das mesmas e que pode ser acessado através de um browser, sendo a tradução e envio do código às placas feito via nuvem própria.

Figura 22 – Editor de texto GNU NANO do Raspberry Pi



Fonte: Autor

Figura 23 – Particle IDE



The screenshot shows the Particle IDE interface with the file 'localtoservo-udp.ino' open. The code is as follows:

```
1 #include <stdlib.h>
2 #include "application.h"
3 int ledPin = D7;
4 SYSTEM_MODE(SEMI_AUTOMATIC);
5 // Set the system mode to semi automatic communication
6 unsigned int localPort = 8888;
7     int angle_x;
8     int angle_y;
9 // An UDP instance to let us send and receive packets over UDP
10 UDP Udp;
11 //TCPClient server = TCPServer(serverPort);
12 //TCPClient client;
13 //IPAddress server[] = { 74, 125, 224, 72 };
14 //LED
15 bool ledState = false;
16 void toggled() {
17     ledState = !ledState;
18     digitalWrite(ledPin, ledState);
19     //delay(70);
20 }
21 //servo control
22 Servo myservo;
23 Servo myservo2;
24
25 void setup() {
26     WiFi.connect();
27     //Particle.disconnect();
28     pinMode(ledPin, OUTPUT);
29     Serial.begin(9600);
30     //WiFi Setup
31
32     // start the UDP
33     Udp.begin(localPort);
34     //servo
35     myservo1.attach(D2);
36     myservo2.attach(D3);
37
38
39
40
41 }
42
43 void loop() {
44     // Particle.disconnect();
45     // going to receive 4 samples from the transmitter
46     // Read first char of data received
}
```

The status bar at the bottom right says 'Ready.'

Fonte: Autor

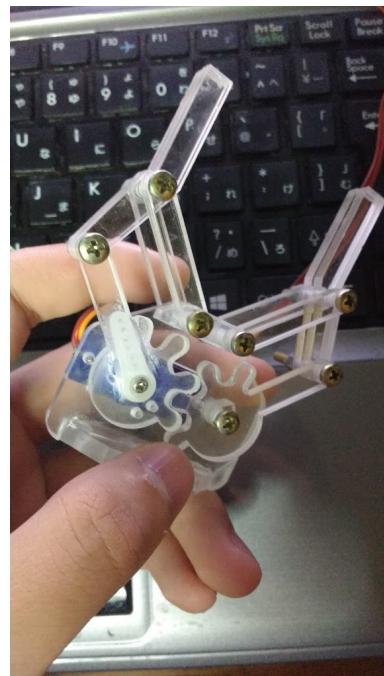
4 Execução e Resultados obtidos

Ete capítulo apresenta as atividades realizadas resultados atingidos, obtidos a partir da especificação do sistema e da metodologia de trabalho adotada.

4.0.1 Controle abre-fecha e rotação da garra

Primeiramente, foi feito o estudo de movimentação garra e de seu controle, uma vez que seu controle é totalmente independente da posição do braço. Para os testes, os comandos foram gerados à partir de uma placa Arduino Mega 2560([Figura 24](#)).

Figura 24 – Garra do braço robótico



Fonte: Autor

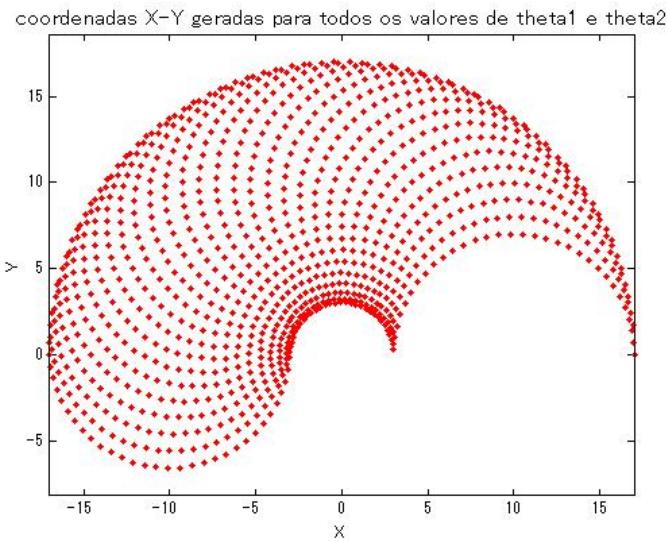
4.0.2 Modelagem de movimentação do braço

Foi feita a modelagem de todos os movimentos possíveis para o braço robótico para posterior uso como referência durante o projeto.

4.0.2.1 Modelagem inicial

Inicialmente, foi feita a modelagem para dois motores (ombro e cotovelo), e gradualmente o modelo foi melhorado([Figura 25](#)).

Figura 25 – Coordenadas X-Y para todos os valores de theta1 e theta2

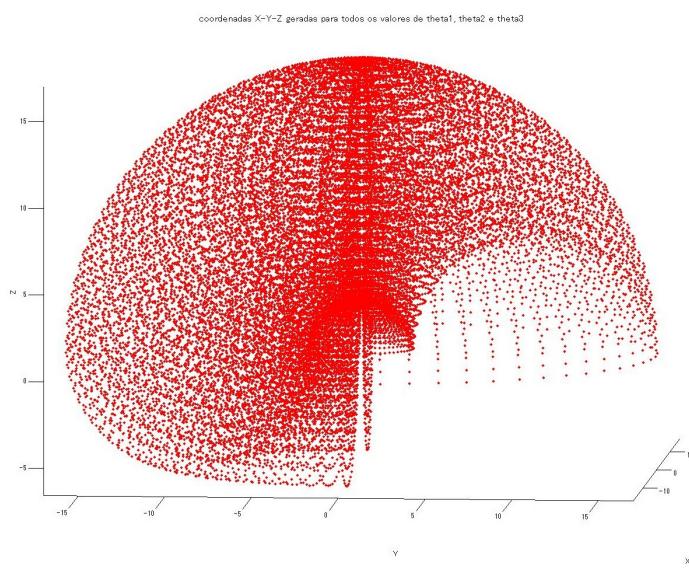


Fonte: Autor

4.0.2.2 Modelagem da movimentação do braço para os motores correspondentes ao ombro, cotovelo e rotação do ombro do braço

O modelo foi ampliado para incluir o terceiro dos 4 motores de movimento do braço(Figura 26).

Figura 26 – Coordenadas X-Y-Z para todos os valores de theta1, theta2 e theta3

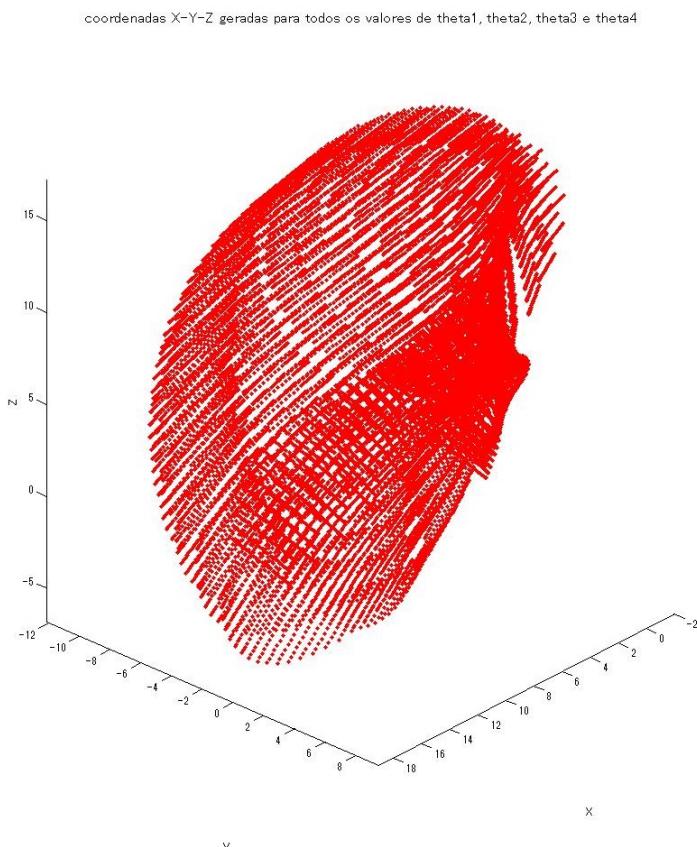


Fonte: Autor

4.0.2.3 Modelagem da movimentação do braço para os motores correspondentes ao ombro, cotovelo, rotação do ombro e pulso do braço

Inclusão do quarto e último motor referente ao braço, completando o modelo a ser utilizado([Figura 27](#)).

Figura 27 – Coordenadas X-Y-Z para todos os valores de theta1, theta2, theta3 e theta4



Fonte: Autor

4.0.3 Aquisição de imagens pela camera

[Figura 17](#): Imagem das cameras estereoscópicas A aquisição das imagens foi feita usando um computador Raspberry Pi rodando o sistema operacional Raspbian Jessie.

Para isso, foram testados os programas motion, gstreamer e Mjpg-Streamer. A seguir, é feita uma comparação dos programas: motion: baixa complexidade de configuração, taxa de captura baixa, alto consumo de capacidade de processamento. Foi logo descartado Gstreamer: alta complexidade para configuração inicial, taxa de captura moderada, alto consumo de capacidade de processamento, não amigável ao uso de duas câmeras.

mjpg-Streamer: Configuração inicial de dificuldade moderada, taxa de captura adequada quando configurado para modo de baixa resolução. Após a escolha do programa, foi feito o estudo para a captura em tempo real de duas câmeras ao mesmo tempo. Para isso, foi primeiro feito o teste para duas instâncias do programa rodando ao mesmo tempo, após ajuste de taxa de captura e resolução provou-se que o programa mantinha-se estável o suficiente para aplicação. (Figura 28).

Figura 28 – Imagem obtida pelas câmeras estereoscópicas usando mjpg-streamer

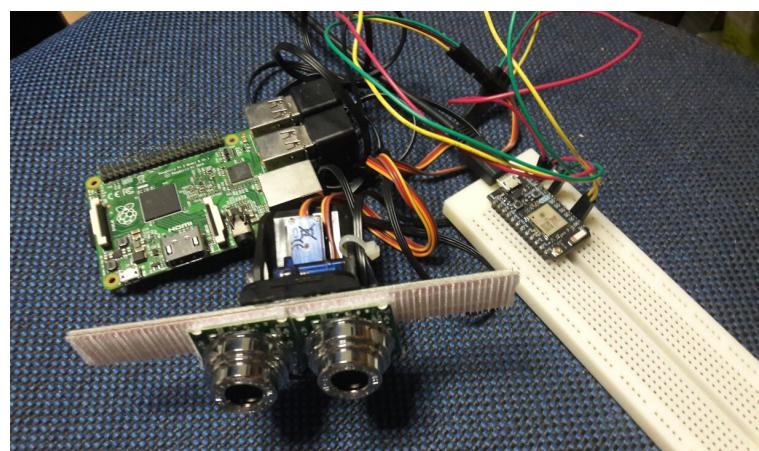


Fonte: Autor

4.0.4 Montagem do sistema de motores e controle dos motores da câmera

Após a configuração das câmeras, foi feita a montagem de sua base com motores, que pode ser vista abaixo(Figura 29):

Figura 29 – Montagem das câmeras estereoscópicas



Fonte: Autor

Após a montagem, foi feita a configuração de duas placas Particle Photon que foram utilizadas como transmissor e receptor, devido ao seu tamanho e peso reduzidos e ao fato de possuir uma antena WiFi já embutida. Para a captura dos movimentos da cabeça do usuário, decidiu-se acoplar um giroscópio externo aos óculos de realidade virtual, de modo que o projeto não dependesse de formas de captura de movimentos específicas para cada modelo de óculos.

Figura 30 – Óculos de realidade virtual com destaque para o giroscópio externo



Fonte: Autor

Para o transmissor, foram realizadas duas versões de código. Inicialmente, foi escrito um código usando-se a saída do giroscópio e realizando as conversão usando-se como base o algoritmo descrito em ([DEBRA](#),). Este código era altamente dependente da posição inicial do giroscópio, que era usada para uma rotina de calibragem inicial.

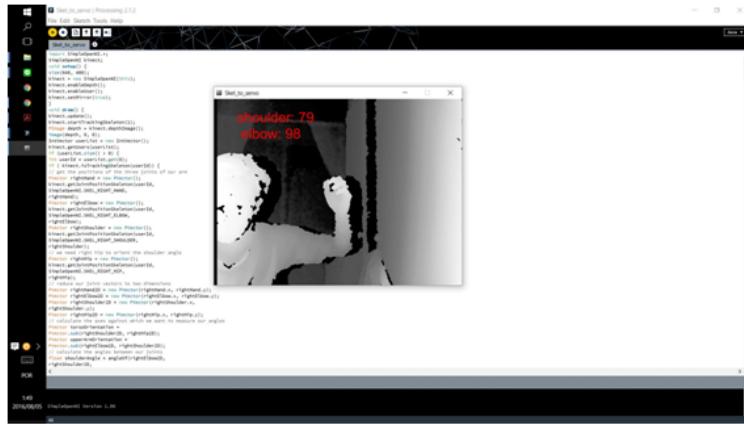
A segunda versão do código foi realizada usando-se bibliotecas específicas ("MPU6050/I2Cdev.h" "MPU6050/MPU6050_6Axis_MotionApps20.h") traduzidas de arduino, que fazem uso de algoritmos internos ao giroscópio para a calibragem , tornando-a mais rápida e precisa

Para o receptor, foi feita uma única versão, responsável por receber os ângulos de rotação e transmiti-los aos dois motores na base da câmera A transmissão dos dados atualmente se dá via protocolo UDP em uma rede local.

4.0.5 Captura de ângulos usando Processing

Foi utilizado inicialmente a linguagem Processing (baseada em Java), usando-se da biblioteca OpenNI para a captura dos movimentos. No entanto, devido a baixa precisão da captura usando a biblioteca, essa abordagem foi abortada.

Figura 31 – Imagem obtida pelas câmeras estereoscópicas



Fonte: Autor

4.0.6 Captura de ângulos usando Visual C#

A programação foi realizada em Visual C# usando-se como base o exemplo de captura e visualização de esqueleto do Microsoft Kinect 1.8 fornecido pela Microsoft, ao qual foram adicionadas funcionalidades necessárias ao projeto. No estado atual, o programa pode além de exibir o esqueleto com as juntas do usuário, capturar os ângulos do braço do usuário e exibi-los como números, para que seja feito o teste de coerência dos dados a serem enviados([Figura 32](#)).

4.0.7 Controle dos servomotores via conexão serial usando Visual C#

Foi feita a integração do controle dos servomotores com o programa responsável pela captura dos movimentos do usuário via conexão serial USB([Figura 33](#)).

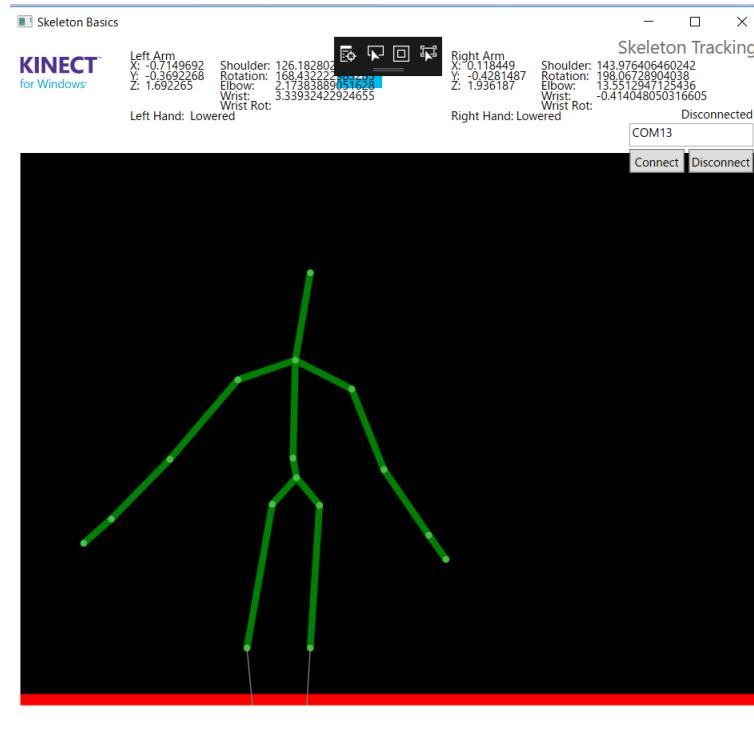
4.0.8 Controle dos servomotores via conexão UDP usando Visual C# e testes de integração

Nesta última etapa, foi feita a tradução do código para a SDK 2.0 do Kinect e a inclusão da funcionalidade WiFi ao programa. Com isso, foi feito de todo o sistema o teste em rede local, em que os atrasos de transmissão foram considerados irrelevantes ($<0.1\text{s}$)[\(??\)](#).

4.1 Resultados

À partir da execução do projeto, foi possível concluir que existe um grande espaço para o desenvolvimento deste tipo de projeto em aplicações futuras.

Figura 32 – Captura de ângulos do braço humano usando Visual C# e Kinect



Fonte: Autor

Figura 33 – Teste do motor da base do braço



Fonte: Autor

Foi constatado que o sistema de câmeras, apesar da baixa resolução do hardware utilizado, é capaz de fornecer uma razoável noção de profundidade, e aliado com a sua capacidade de seguir a movimentação do usuário, tem aplicações comerciais imediatas.

Percebeu-se no entanto, que existe uma baixa resistência a ruídos por parte do

Figura 34 – Teste de integração



Fonte: Autor

sensor óptico Microsoft Kinect que apresentava altas taxas de erro de leitura em testes com mais de um ser humano em tela.

É relevante notar também que o mesmo apresentou dificuldade na leitura do usuário quando este se encontrava de perfil em relação ao sensor, o que, aliado a falta de noção espacial do usuário em relação ao próprio corpo quando no modo de imersão, pode causar dificuldades na interação com o ambiente remoto, com o possível descontrole do braço.

É possível assim, concluir que ainda são necessárias melhorias antes de tal abordagem ser aplicável a usos comerciais.

5 Conclusão

O sistema desenvolvido permitiu a interação do usuário com um ambiente através da coleta de dados de movimento tanto por tecnologia vestível como por tecnologia sem marcadores e a imersão visual estereoscópica neste ambiente, cumprindo com grande parte das especificações do projeto.

Entretanto, o escopo do trabalho proposto inicialmente foi reduzido para entrega deste documento. Assim, os testes que antes seriam realizados com o sistema conectado à internet foram executados somente em rede local.

Isto pode ser explicado pelos desafios enfrentados durante o desenvolvimento do projeto, tais como:

- Utilização de tecnologias diferentes: cada componente foi desenvolvido em plataformas e linguagens distintas, o que demandou um considerável tempo de familiarização e estudo. Além disso, houveram mudanças de hardware durante o projeto, o que exigiu que alguns códigos tivessem que ser mudados
- Integração dos componentes do sistema: a comunicação entre os componentes, embora utilize técnicas e protocolos existentes, foi consideravelmente trabalhosa e de difícil depuração.
- Ausência de certos componentes no mercado brasileiro: Parte do hardware teve que ser importada ou construída especialmente para o projeto, tirando o foco temporariamente da programação e causando atrasos generalizados no projeto.

Algumas sugestões de trabalhos futuros para aprimoramento deste projeto são:

- Identificação e seleção do usuário a operar o sistema;
- Gravação dos dados de movimento do usuário para execução de tarefas repetitivas;
- Uso de motores com feedback da posição atual para medida de precisão e implementação de algoritmos de controle;
- Armazenamento das imagens obtidas pelas câmeras;
- Adaptação do código a outros tipos de atuadores;
- Adição de um sistema de locomoção para o sistema;
- Configuração de um servidor conectado à internet para aumento da distância de operação;

- Adição de conexão GSM, para controle independente de roteadores locais.
- Melhorias de hardware/software para melhora de precisão;

Finalmente, foi possível concluir que existe uma grande possibilidade no desenvolvimento para o método de operação de braço robótico estudado em aplicações futuras, mas que no entanto, no estado atual da tecnologia, usos comerciais ainda se provam inviáveis devido a alta incidência de erros de leitura. Foi possível concluir também, que a estereoscopia pode ter aplicações imediatas, com grandes possibilidades de desenvolvimento para um futuro próximo

Comentários finais

Devido ao caráter interdisciplinar deste projeto, não só foi possível como necessária a aquisição de muitos conhecimentos, técnicas e termos da engenharia de computação, sendo todo o processo do projeto marcado pelo aprendizado.

Tal característica do projeto reflete o ideal engenheiro de busca pelo conhecimento que permeia todo o curso na Escola Politécnica.

Finalmente, julgo como satisfatória e gratificante a experiência de poder realizar todas as etapas de um projeto de engenharia, desde sua concepção até sua conclusão e análise dos resultados, podendo observar em primeiro plano a execução e aplicação de conceitos teóricos vistos (e não vistos) em aula.

Referências

DEBRA. Citado na página 40.

ALEOTTI, J.; SKOGLUND, A.; DUCKETT, T. Gesture controlled mobile robotic arm using accelerometer. *International Journal of Innovative Research in Science, Engineering and Technology*, v. 4. Disponível em: <<http://aass.oru.se/~asd/Aleotti04IMG.pdf>>. Citado na página 23.

BRUMSON, B. *Robotics in Security and Military Applications*. Robotic Industries Associaton, 2011. Disponível em: <http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotics-in-Security-and-Military-Applications/content_id/3112>. Acesso em: 6 mar 2016. Citado na página 23.

CSA. *Uses for Robotic Arm Technology*. Canadian Space Agency, 2011. Disponível em: <<http://www.asc-csa.gc.ca/eng/canadarm/robotic.asp>>. Acesso em: 22 abr 2016. Citado na página 23.

DEBRA. *Gyroscopes and Accelerometers on a Chip*. Disponível em: <<http://www.geekmomprojects.com/gyroscopes-and-accelerometers-on-a-chip/>>. Acesso em: 6 mar 2016. Citado na página 57.

DU, G. et al. Markerless kinect-based hand tracking for robot teleoperation. IEEE, v. 33, 2012. Disponível em: <http://cdn.intechopen.com/pdfs/37895/InTech-Markerless_kinect_based_hand_tracking_for_robot_teleoperation.pdf>. Acesso em: 6 mar 2016. Citado na página 23.

FOUNDATION, R. P. *Raspberry Pi main page*. Disponível em: <<https://www.raspberrypi.org/>>. Acesso em: 6 mar 2016. Citado na página 42.

FOUNDATION, T. A. *Apache HTTP Server Project Main Site*. Disponível em: <<https://httpd.apache.org/>>. Acesso em: 6 mar 2016. Citado na página 46.

ICOR. *CALIBER MK4 LARGE EOD ROBOT Datasheet*. [S.l.], 2016. Disponível em: <<https://icor-b23.kxcdn.com/wp-content/uploads/2016/09/CALIBER%C2%AE-MK4-V2016.08.25.pdf>>. Acesso em: 16 nov 2016. Citado na página 23.

JPL. *NASA's JPL maneuvers a robot arm with Oculus Rift and Kinect 2, points to more immersive space missions*. Disponível em: <<http://www.hi.jpl.nasa.gov/projects/>>. Acesso em: 6 mar 2016. Citado na página 24.

MICROSOFT. *Microsoft Kinect main page*. Disponível em: <<https://developer.microsoft.com/en-us/windows/kinect>>. Acesso em: 6 mar 2016. Citado na página 39.

MICROSOFT. *Visual C sharp*. Disponível em: <<https://msdn.microsoft.com/pt-BR/library/kx37x362.aspx>>. Acesso em: 6 mar 2016. Citado na página 44.

IMITSANTISUK, C.; KATSURA, S.; OHISHI, K. Sensorless interaction force control based on b-spline function for human-robot systems. *SICE Journal of Control, Measurement, and System Integration*, v. 1. Disponível em: <https://www.jstage.jst.go.jp/article/jcensi/1/6/1_6_452/_pdf>. Citado na página 23.

ORACLE. *Java main page*. Disponível em: <<https://www.java.com/en/>>. Acesso em: 6 mar 2016. Citado na página 43.

PARTICLE. *Particle main page*. Disponível em: <<http://particle.io>>. Acesso em: 6 mar 2016. Citado na página 40.

STÖVEKEN, T. *mjpg-streamer repository*. Disponível em: <<https://github.com/jacksonliam/mjpg-streamer>>. Acesso em: 6 mar 2016. Citado na página 46.

WANG, Y. et al. Robot teleoperation using a vision-based manipulation method. IEEE, v. 33, 2005. Disponível em: <<http://ieeexplore.ieee.org/abstract/document/5685167/>>. Acesso em: 6 mar 2016. Citado na página 23.