

Sistema de Interação por Gestos em Ambientes Inteligentes

Eduardo Freitas Goulart¹, Reiner Franthesco Perozzo¹

¹Curso Ciência da Computação – Centro Universitário Franciscano (UNIFRA)
97.010-491 – Santa Maria – RS – Brasil

edufreitasgoulart@hotmail.com, reiner.perozzo@unifra.br

Abstract. *Intelligent environments are automation scenarios that react according to user behavior and characterized also by beholding multimodal interaction strategy. This work proposes the development of an interaction system based on gesture recognition and facing the building / home automation using the Kinect to identify the gestures and the Arduino to the automation environment.*

Resumo. *Os ambientes inteligentes são cenários de automação que reagem de acordo com o comportamento do usuário e caracterizam-se, também, por contemplarem estratégia de interação multimodal. Assim, este trabalho propõe o desenvolvimento de um sistema de interação baseado no reconhecimento de gestos e voltado para a automação predial/residencial utilizando o Kinect para identificação dos gestos e o Arduino para a automação do ambiente.*

1.Introdução

Os ambientes inteligentes são cenários automatizados, controlados por computador, capazes de reagir de maneira inteligente à presença de pessoas, atendendo suas necessidades [Guerra 2007]. Esses ambientes inteligentes envolvem conceitos de computação ubíqua e pervasiva. Toda a computação envolvida no ambiente é imperceptível ao usuário, através de uma interface de interação [Ferraz e Messias 2011]. Esses ambientes são automatizados e providos de tecnologias direcionadas tanto para a comodidade quanto a saúde e a gestão de recursos [Souza, Correa e Machado 2013].

Há diversas formas de interação para esses ambientes, tais como: *displays, smartphones, mp3, vídeo games*, entre outros. Uma dessas possibilidades de formas de interação é o reconhecimento de gestos, que vem se tornando bastante popular principalmente na tecnologia de *games*, dispensando os controles de botões e fios elétricos. Uma das plataformas que usa essa tecnologia é o Xbox, da Microsoft, que através do sensor Kinect rastreia o esqueleto humano, e conseqüentemente, reconhece os movimentos [Microsoft 2016]. Sendo que esse é um dos aspectos abordados neste trabalho.

Uma interface baseada em gestos tem a sua estrutura podendo ser dividida em três partes, sendo elas: segmentação, reconhecimento e interpretação. A primeira, segmentação, realiza a detecção da pessoa na cena e transforma a sua pose num determinado momento. A interpretação das informações obtidas na etapa de segmentação é o reconhecimento, que ainda identifica o gesto sendo realizado e a interpretação contextualizará uma sequência de gestos ou o gesto, extraindo desta uma atividade de interação [Ghirotti e Morimoto 2010].

Diante desse contexto, o presente trabalho tem como proposta desenvolver um sistema de interação por gesto em ambientes inteligentes, usufruindo dos seus conceitos e tecnologias.

1.1. Objetivos Gerais

Desenvolver um sistema de interação em ambientes inteligentes, utilizando o Kinect para identificação dos gestos e o Arduino para o controle da automação do ambiente.

1.2. Objetivos Específicos

- Desenvolver um sistema de interação por gestos que permita a interação do usuário com ambientes inteligentes;
- Construir um cenário que permita a execução do sistema;
- Permitir conexão com o microcontrolador;
- Mapear gestos e transformá-los em comandos.

2. Referencial Teórico

Esta seção tem como objetivo abordar a revisão bibliográfica contendo conceitos, tecnologias e trabalhos relacionados, que servem como apoio para elaboração da proposta deste trabalho.

2.1. Revisão Bibliográfica

Nesta subseção serão apresentados os Ambientes Inteligentes, a Interação Humano-Computador, o Kinect e os microcontroladores, cujo conceitos e tecnologias são consideradas para a elaboração da proposta e, conseqüentemente, o desenvolvimento do trabalho.

2.1.1. Ambientes Inteligentes

Os avanços tecnológicos da indústria eletrônica, têm sido alcançados pela miniaturização das dimensões de dispositivos [Augusto, Nakashima e Aghajan 2009]. Entende-se por ambientes inteligentes cenários automatizados que regem o comportamento do usuário. Essa definição inclui a necessidade de um sistema sensível ao contexto, também conhecido como *context-aware*. Esse termo, ambientes inteligentes, engloba uma infraestrutura física que inclui sensores, atuadores e redes de comunicação de dados que integra todos esses elementos [Augusto, Nakashima e Aghajan 2009]. O objetivo dos ambientes inteligentes é tornar transparente para as pessoas a interação com sistemas computacionais em um determinado cenário de aplicação [Guerra 2007].

Um dos possíveis cenários de aplicação dos ambientes inteligentes é na Domótica, como a automação predial/residencial [Peçanha 2012]. Tal automação visa auxiliar nas tarefas diárias dos moradores os quais podem evitar preocupações como, por exemplo, esquecer as janelas abertas quando a previsão for chuva [Bolzani 2004]. A automação residencial é um mecanismo de satisfazer as necessidades básicas de comunicação, segurança, gestão energética e conforto de uma habitação. Ela se caracteriza pelo conjunto de serviços proporcionados por sistemas tecnológicos interligados. O fator principal que define uma residência automatizada é a integração

entre os sistemas, juntamente com a capacidade de executar funções e comandos diante de instruções programáveis [Muratori e Dal Bó 2011].

A estrutura física desses ambientes inteligentes é formada por sensores e atuadores. Os sensores são dispositivos que detectam um sinal originado de um estímulo externo podendo ser de indicação direta, como termômetros de mercúrio, ou em par com um indicador a partir de sensores analógicos e digitais, providos de *display*. Existem alguns tipos de sensores como, por exemplo, os sensores de temperatura (que são os mais utilizados nos projetos de automação), os sensores de incêndio (que são utilizados na segurança) e, também, existem sensores de umidade e de presença [Pistolozzi 2009]. Os dispositivos atuadores caracterizam-se por modificar uma variável controlada, agem sobre um sistema controlado e recebem um sinal proveniente do controlador. São exemplos de atuadores as válvulas (pneumáticas e hidráulicas), relés, cilindros (pneumáticos e hidráulicos), motores, solenoides, entre outros [Wendling 2010].

2.1.2. Interação Humano-Computador (IHC)

A IHC é uma área em crescente desenvolvimento tendo característica multidisciplinar e apresenta o objetivo de tornar máquinas sofisticadas mais acessíveis, no que se refere a interação aos seus usuários potenciais [Carvalho 2003].

Existe uma ampla variedade de sistemas interativos. Um exemplo famoso é o site chamado *Facebook*, que permite às pessoas interagirem com seus amigos, compartilhando fotos digitais, vídeos, notícias, aplicativos, bate-papo e, assim, formando uma rede social. Outros exemplos de sistemas interativos estão nos brinquedos, em que o uso da robótica contemplando entrada e saída de voz e uma variedade de sensores, permitem interações diversas, afim de intensificar o conhecimento de crianças ao brincar [Benyon 2011].

As interações multimodais permitem que o usuário utilize diversos modos de comunicação, interagindo com o sistema a partir de toque, voz, movimentos e gestos. A utilização dos modos pode ser em sequência ou simultaneamente e em combinação ou independentemente. Além da entrada de teclado e *mouse* e saída através de uma interface gráfica. A relação existente entre um dispositivo de entrada ou saída (microfone, teclado, tela sensível ao toque) e uma linguagem de interação (linguagem natural, manipulação direta) chama-se modalidade. Definindo, assim, a interação multimodal como a utilização de duas ou mais modalidades para interagir com um sistema [Talarico Neto 2011].

Um outro exemplo são os sistemas *multitouch*, que permitem aos usuários operar seus computadores além da utilização de equipamentos convencionais (*mouse* e teclado). Esse sistema consiste em uma tela para o uso do toque com dedos ou outros objetos postos sobre a superfície, em que é executado um *software* para reconhecimento dos pontos de contato sobre a superfície. Atualmente, essa técnica é usada principalmente em *tablets* e *smartphones* [Campos 2009].

2.1.3. Kinect

Ao longo dos anos cresce cada vez mais propostas inovadoras de interação com o usuário, focadas no paradigma da interface natural. A primeira interface foi chamada de *Command Line Interface*, em que o usuário utilizava comandos de entradas textuais que

desempenhavam funções de um aplicativo. Ao passar do tempo surgiu a interface chamada *Graphical User Interface*, que faz o uso de janelas gráficas, *mouse*, e outros componentes visuais. Esse conceito é largamente utilizado em sistemas operacionais. Com o aumento de usuários utilizando a computação surgiu, então, o conceito de interfaces naturais, a qual está baseada no uso de linguagem natural como gestos e comando de voz. Resultando no lançamento do Kinect, pela Microsoft. [Cardoso 2014].

O Kinect foi lançado, oficialmente, em 2010, como sendo (a princípio) um acessório para o *videogame* Xbox 360. Em 2011 a Microsoft anunciou o lançamento do Kinect versão Windows, juntamente com um *Software Development Kit* (SDK), permitindo, assim, que os desenvolvedores de *software* pudessem usar todas as funcionalidades do Kinect em suas aplicações [Catuhe 2012].

Os desenvolvedores que usam o Kinect podem optar entre o uso das linguagens C++, C# e Visual Basic. Atualmente existem 4 tipos de Kinect diferenciados pelos nomes: Kinect, Kinect for Windows, Kinect for Xbox 360 e Kinect for Xbox One. O Kinect for Windows possui microfones de melhor qualidade (se comparados aos outros modelos) e é capaz de rastrear o usuário mais próximo ou até na posição sentado. A diferença do Xbox 360 para os outros está no cabo específico, que é ligado a um adaptador e na energia elétrica, sendo possível conectá-lo ao computador [Cardoso 2014].

A estrutura do Kinect é composta por uma câmera *Red Green and Blue* (RGB), responsável por armazenar três dados do canal com uma resolução 1280x960 pixels, fazendo com que seja possível a imagem ficar colorida. Além disso, ainda faz parte um emissor *Infrared* (IR) e um sensor de profundidade IR. O sensor de profundidade lê os feixes de IR refletida de volta para o sensor e o emissor emite feixes de luz infravermelha. Os feixes refletidos são convertidos em informação de profundidade da medição da distância entre um objeto e o sensor, fazendo com que seja possível a captura de uma imagem de profundidade. Faz parte de sua composição, também, um conjunto contendo quatro microfones para captar o som, sendo possível gravar áudio e ainda encontrar a localização da fonte sonora com a direção da onda de áudio. Apresenta, ainda, um acelerômetro de três eixos configurado para uma gama 2G, em que G representa a aceleração da gravidade. E, com isso, é possível utilizar o acelerômetro para determinar a orientação do Kinect [Microsoft 2012].

O rastreamento feito pelo Kinect permite uma representação aproximada das diferentes partes do corpo humano e suas articulações. Cada movimento realizado pelas articulações é obtido individualmente, permitindo, então, a tradução dos gestos definidos para determinado sistema [Oliveira 2014]. Ao total, são rastreadas vinte *joints* (articulações) e suas respectivas posições X, Y e Z. O Kinect tem a capacidade de identificar até seis pessoas, sendo que somente em duas ele reconhecerá o esqueleto completo [Moreira 2013].

2.1.4. Microcontroladores

Segundo Souza (2005) um microcontrolador pode ser definido como um componente eletrônico, cuja “inteligência” é programável. Ele é utilizado no controle de processos lógicos. Em outras palavras, um microcontrolador pode ser considerado um computador em um único *chip*, contendo processador, memória, mecanismos de entrada e saída de dados, temporizadores entre outros [Penido e Trindade 2013].

Com base em microcontroladores foram desenvolvidas plataformas eletroeletrônicas que buscam facilitar e agilizar a prototipação de sistemas embarcados. Uma delas chama-se Arduino [Arduino 2016] que é baseada no microcontrolador Atmel. O Arduino é uma plataforma capaz de ler dados de entrada (como sensores) e transformá-los em saída de dados (acionamentos de leds e servos motores, por exemplo). O Arduino possui uma *Integrated Development Environment* (IDE) onde é possível programar a plataforma em uma linguagem de programação, também chamada Arduino. Esta, por sua vez, é baseada em C / C++. Toda a plataforma Arduino é *open-source/ open-hardware* e, por ser uma plataforma aberta, possui vários modelos que são diferenciados pela quantidade de memória, pinos de entrada e saída e outras características [Arduino 2016]. A plataforma Arduino pode ser usada em diversas áreas, entre elas a de automação predial/residencial e a robótica.

2.2. Trabalhos Relacionados

Esta seção apresenta três propostas que estão relacionadas com este trabalho. São descritas três abordagens que utilizam sistemas de interação humano-computador para empregar as tecnologias disponíveis e criar outros meios de melhor usufruir estes sistemas interativos.

2.2.1. Protótipo de uma interface homem-computador baseada em interação por voz, gestos, multitoque e transparência para automação predial

Esse trabalho [Oliveira 2014] apresenta um protótipo chamado inOctopus, cujo nome é resultado de uma combinação entre os termos interface natural (*in*) e braços independentes (*Octopus*). Essa proposta está centrada no controle de várias interações multimodais combinadas. A interface homem-computador do protótipo permite uma interação por voz, gestos, multitoque e transparência, baseados em redes sem fio, Kinect e Arduino. No desenvolvimento do trabalho, há combinações de tecnologias, permitindo que a aplicação consiga controlar os recursos e dispositivos de automação de prédios, suprimindo a necessidade de conforto e mantendo a produtividade dos usuários que destes recursos utilizarem.

Para a construção do protótipo que valida a proposta tem-se o Kinect (como *hardware* de reconhecimento de gestos e de voz) e o Arduino (como plataforma para gerenciamento da automação). A comunicação entre o inOctopus e o Arduino é realizada por meio de *Transfer / Internet Protocol* (TCP/IP). Ao executar o inOctopus há uma interface *multitouch* que contém os ícones que estão associados às funcionalidades como, por exemplo: acionamento de lâmpadas, informar a temperatura e o controle do ar condicionado, permitir o controle dos dispositivos de *Datashow*, câmera de segurança, exibir a temperatura média do ambiente, acionar os sensores de presença, ativar e desativar o reconhecimento de voz, controlar o som ambiente, ativar e desativar o controle automático de temperatura e, por fim, permitir o acionamento do modo econômico admitindo apenas uma lâmpada e um ar condicionado ligado. Na interface, quanto ao uso de gestos, o inOctopus está baseado em gestos das mãos direita, esquerda e o gesto aceno, para gerenciar uma apresentação na ferramenta Microsoft PowerPoint. Já, para a opção de comando de voz, é possível enviar comandos para um condicionador de ar, sistema de iluminação, sensores de presença, projetores e sistema de som do ambiente.

2.2.2. Sistema de controle residencial baseado na plataforma Arduino

Nesse trabalho [Silva e Candido 2011] é proposto o desenvolvimento de um sistema de controle residencial, em que através de uma aplicação de *smartphone* é possível controlar o sistema de iluminação e o sistema de segurança. Na solução foi construído um circuito elétrico e programada uma aplicação para o Arduino. Além disso, foi desenvolvido um aplicativo para gerenciamento em *smartphones* com sistema operacional Android.

Para realizar uma simulação do ambiente residencial foi utilizado uma maquete com três cômodos e um banheiro. No cenário, lâmpadas foram representados por LEDs e com os emissores e receptores de infravermelho foram construídos os sensores de presença. A comunicação entre o Arduino e a aplicação é feita via *Bluetooth*.

Foi construído, então, um ambiente de teste da aplicação utilizando uma comunicação serial para interligar o Arduino e o computador. Em nível de programação, a aplicação foi implementada na linguagem Java e para a comunicação serial foi utilizada a biblioteca RXTXcomm.

A função da aplicação é coletar uma entrada de dados que é digitada pelo usuário e enviá-la, pela porta serial, ao Arduino. Para identificar a ação escolhida pelo usuário foi definido um conjunto de constantes, sendo que esses valores devem ser os mesmos tanto no Arduino quanto no Android. O sistema de iluminação oferece controle sobre todas as lâmpadas que estão ligadas nas portas digitais do Arduino. Na parte de segurança, o sistema possui sensores de presença e uma sirene, a qual é ativada na detecção de presença. Na tela de interação, ao selecionar um dos cômodos o usuário é redirecionado para uma tela de controle que captura os comandos, enviando-os para o Arduino. Além do controle individual dos cômodos, o sistema oferece a tela de controle geral, que possibilita habilitar ou desabilitar todos os componentes que fazem parte do projeto.

2.2.3. Sistema interativo baseado em gestos para utilização de comandos no computador

Esse trabalho [Almeida 2013] propõe o projeto de um sistema interativo controlado pela detecção de gestos através do sensor Kinect, a fim de poder realizar comandos no sistema operacional do computador pessoal.

A comunicação do Kinect e do sistema interativo são realizadas utilizando SDK fornecido pela Microsoft. O protótipo da interface do trabalho divide-se em uma tela de gestos básicos, outra tela de gestos específicos, uma tela de gestos avançados e ainda a tela de configurações, sendo que, nessa última é possível ajustar a angulação e distância. De acordo com o movimento feito, é executado um comando ao sistema operacional. Por exemplo, se usuário deseja clicar, então, deve mover o braço para frente. Assim, de acordo com cada gesto há um comando associado para realizar ações no sistema operacional.

2.2.4. Considerações sobre os trabalhos relacionados

Ao realizar o estudo dos trabalhos correlatos, pode-se perceber algumas semelhanças, tanto no enfoque conceitual quanto nas tecnologias usadas. O primeiro e o terceiro trabalhos descritos se assemelham-se com esta proposta por apresentarem interação por

gestos usando Kinect nas suas aplicações. No segundo foi construído um outro modelo de sistema interativo, aplicando o uso do Arduino e *smartphone* no controle de uma residência. Todas as aplicações oferecem protótipos de interfaces, que possibilitam acessibilidade ao usuário. Diferentemente do primeiro trabalho relacionado, que desenvolve um sistema de gestos voltado para a manipulação do Microsoft PowerPoint, este projeto visa implementar um sistema baseado em gestos, porém direcionados aos dispositivos acionados pelo microcontrolador.

3. Proposta

Os ambientes inteligentes, além de permitirem conforto e praticidade aos seus ocupantes, também podem auxiliar pessoas com necessidades especiais em tarefas cotidianas. Seja qual for o domínio de aplicação, os sistemas de interação multimodal recebem destaque nesses cenários pelo fato de que eles oferecerem diversas possibilidades os moradores se relacionarem com o ambiente. Sendo assim, este trabalho propõe o desenvolvimento de um sistema de interação para ambientes inteligentes. Essa interação será realizada utilizando o Kinect para identificação dos gestos, o qual terá uma interface de comunicação com um microcontrolador responsável pela automação do ambiente. Em síntese, o ambiente inteligente detecta determinados gestos de seus ocupantes e realiza ações de automação que correspondam aos gestos sinalizados.

A Figura 1 apresenta a visão geral de como são realizados os dois fluxos possíveis para utilização do sistema: i) fluxo de interação e (ii) fluxo de configuração do sistema. No primeiro, o usuário realiza algum gesto (pré-definido na codificação da aplicação) o sensor Kinect reconhece o esqueleto e seus dados, enviando-os para a aplicação, através de uma *Dynamic Link Library DLL* (Kinect.dll), que está disponível no SDK versão 1.8. Essa DLL envia comandos para o sensor, que retorna dados para aplicação através de um adaptador *Universal Serial Bus* (USB) conectado ao computador. A aplicação identifica esses gestos e envia uma mensagem via serial ao microcontrolador, que reconhece a mensagem indicando qual ação ocorrerá no ambiente.

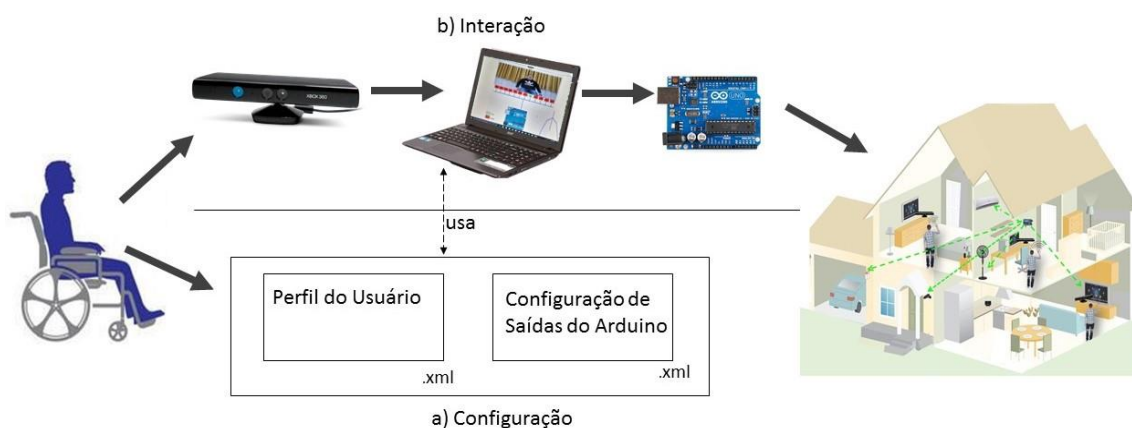


Figura 1 – Interação / Configuração

3.1. Projeto

A construção do trabalho está baseada em *Feature Driven Development* (FDD), por ser uma metodologia ágil que se baseia em processos que podem ser repetidos e bem definidos [Spinola 2010]. O FDD se divide em etapas como o desenvolvimento de um modelo abrangente, a construção de uma lista de funcionalidades, com planejamento, projeto e desenvolvimento por funcionalidades, contemplando a realização dos trabalhos finais de graduação um e dois. No primeiro processo é construído um modelo conceitual que, por meio de um estudo sobre o escopo do sistema e seu contexto, serve para definir e detalhar cada área que será modelada

Considerando o primeiro processo da metodologia utilizada nesse trabalho, é apresentado, na Figura 2, o diagrama de classes do projeto contendo seus respectivos atributos e métodos. A classe principal que é chamada de “Gerenciador” realiza a montagem do esqueleto do usuário para a visualização na interface gráfica e o reconhecimento dos gestos realizados, que associam a um determinado comando para o microcontrolador, ou seja: essa classe é responsável por coordenar todo o sistema. Outra classe a ser destacada é a “Arduino”, que é encarregada por identificar os pinos escolhidos (de entrada e saída presentes na plataforma Arduino) pelo usuário e enviar a mensagem ao microcontrolador.

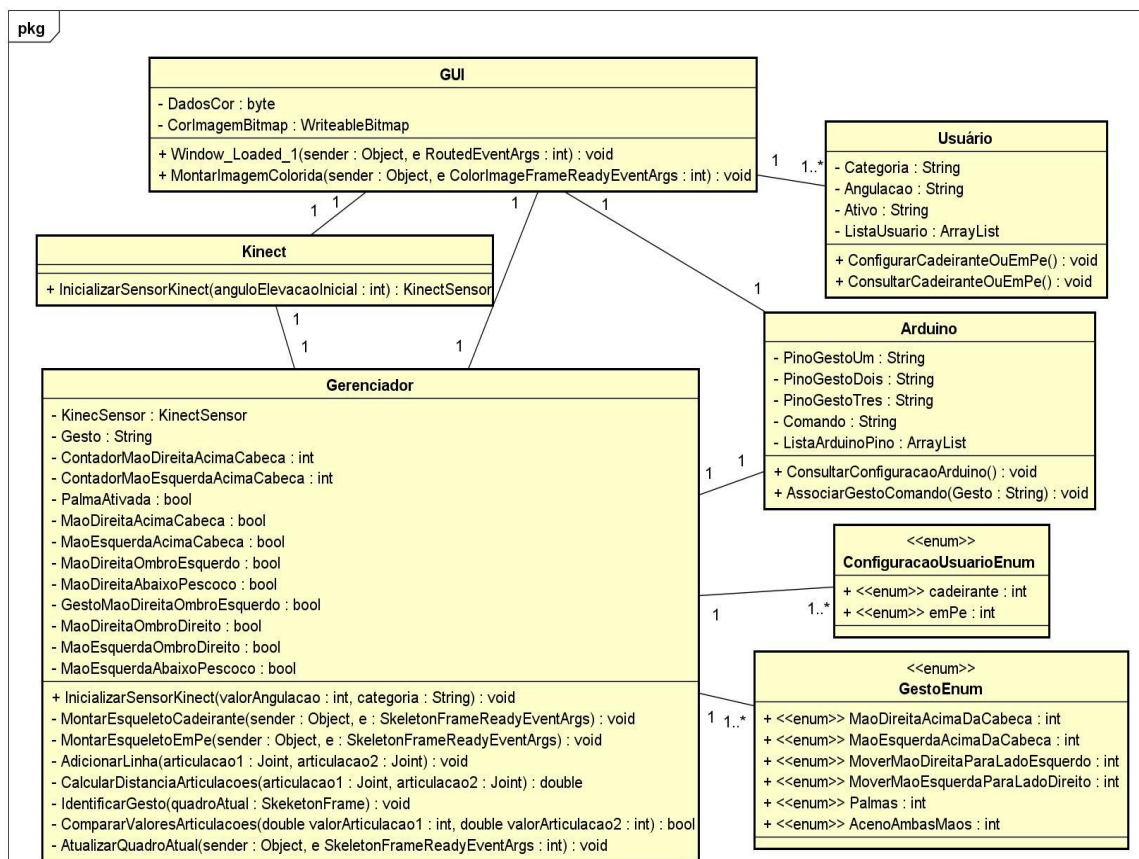


Figura 2 - Diagrama de classes

As trocas de mensagens entre as classes estão representadas por meio de um diagrama de sequência, disponível no Apêndice 1. Nele, consta a ordem de

comunicação entre todos os componentes da aplicação, numeradas (sequencialmente) e estando descritas no Apêndice 2.

No que tange os gestos possíveis a serem realizados pelo usuário na comunicação com a aplicação, eles foram construídos e estão listados na Tabela 1. Tais gestos são representados por pseudo-esqueletos referentes aos membros superiores do corpo humano. Um exemplo de como o usuário realiza um gesto e esse mesmo gesto é identificado pelo sistema é apresentado na Figura 3, em que há uma comparação entre o pseudo-esqueleto, representado na Tabela 1, com o corpo do usuário.

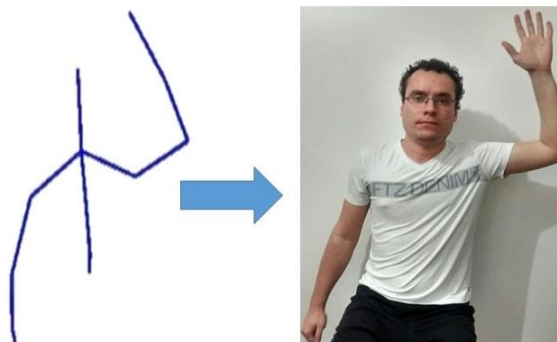
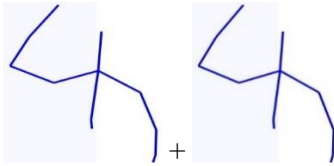
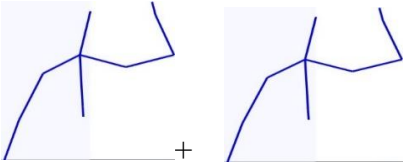


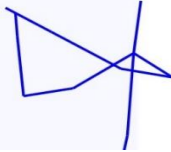



Figura 3 - Pseudo-esqueleto representa o corpo do usuário

Tabela 1 - Tabela de Gestos

GESTOS REALIZADOS	
PinoGestoUm	
	
Mão esquerda duas vezes acima da cabeça	Mão direita duas vezes acima da cabeça
PinoGestoDois	
	
Movimentar a mão direita para lado esquerdo	Movimentar mão esquerda para lado direito
PinoGestoTres	
	
“Bater Palma”	Acenar ambas as mãos

3.2. Implementação

A parte de implementação da proposta, considera a segunda fase da metodologia FDD e contempla as duas etapas apresentadas na Figura 1: a) configuração e b) interação.

a) Configuração

Para a configuração do sistema são utilizados dois arquivos (Figuras 4 e 5) do tipo *xml* (*eXtensible Markup Language*), em que o usuário (ao editá-los), configura o comportamento do sistema, de acordo com suas necessidades. A aplicação necessita consultar estes arquivos para tomada de decisões no momento da interação. Dessa forma, é implementada uma consulta *LINQ* (*Language Integration Query*) para cada *xml*. Na configuração do usuário o método “ConsultarCadeiranteOuEmPe”, realiza a busca através do usuário onde a *tag* ativo contém o valor “sim”, e adiciona esses dados em uma lista. Com usuário incluído na lista o sistema pode saber se a aplicação está configurada para “em pé” ou “cadeirante”. No método “ConfiguraCadeiranteOuEmPe” é validado o valor da angulação do sensor definido no arquivo. Como o limite do ângulo de elevação do sensor Kinect está entre -27 e 27 [Cardoso 2014], na aplicação é definido um limite de -10 e 20 de elevação, caso contrário o usuário deverá editar este valor novamente no arquivo.

```
<?xml version="1.0" encoding="utf-8"?>
<usuarios>
  <usuario id="1">
    <categoria>cadeirante</categoria>
    <!-- Configure a angulação do sensor no mínimo -10 e no máximo 20 -->
    <angulacao>preencher aqui a angulação</angulacao>
    <!-- Configure sim para ativar ou nao para desativar modo cadeirante-->
    <ativo>preencher aqui o ativo </ativo>
  </usuario>
  <usuario id="2">
    <categoria>emPe</categoria>
    <!-- Configure a angulação do sensor no mínimo -10 e no máximo 20 -->
    <angulacao>preencher aqui a angulação</angulacao>
    <!-- Configure sim para ativar ou nao para desativar modo em Pé-->
    <ativo>preencher aqui o ativo</ativo>
  </usuario>
</usuarios>
```

ConfiguracaoUsuario.xml

Figura 4 -Configuração de arquivos de perfil: Configuração Usuário

Na configuração do *xml* (ConfiguracaoArduino.xml) do Arduino, apresentado na Figura 5, o usuário configura os pinos com valores inteiros entre 2 e 13, a fim de que estes valores possam ser concatenados com o comando do tipo *String*, para o envio ao Arduino.

```
<?xml version="1.0" encoding="utf-8"?>
<arduino>
  <!-- Configure os pinos do microcontrolador de acordo com o gesto definido-->
  <!-- Gesto mão esquerda duas vezes acima da cabeça para DESLIGAR/ABRIR-->
  <!--e gesto mão direita duas vezes acima da cabeça para LIGAR/FECHAR -->
  <pinoGestoUm>preencher aqui o numero do pino para o gesto um</pinoGestoUm>
  <!-- Gesto movimentar mão direita para lado esquerdo para LIGAR-->
  <!--e movimentar mão esquerda para direita para DESLIGAR-->
  <pinoGestoDois>preencher aqui o numero do pino para o gesto dois</pinoGestoDois>
  <!-- Gesto palmas para LIGAR e aceno de ambas as mãos para DESLIGAR-->
  <pinoGestoTres>preencher aqui o numero do pino para o gesto tres</pinoGestoTres>
</arduino>
```

ConfiguracaoArduino.xml

Figura 5 - Configuração de arquivos de perfil: Configuração Arduino

b) Interação

A conexão do Kinect com a aplicação é realizada através de uma *dll*, para isso é necessário importar a referência (Microsoft.Kinect), para ter acesso aos componentes e seus dados. O método “InicializarSensorKinect” verifica o status do sensor Kinect. Se conectado, inicia seus componentes de sensores, câmera, quadro de esqueleto, configuração da angulação e retorno de um variável tipo “KinectSensor”.

Na Figura 6 estão representados os componentes da interface gráfica, em que após a inicialização do sensor é possível montar a imagem da câmera *RGB* na interface gráfica. No que tange a representação da interface gráfica, ela é implementada na linguagem *XAML* (*eXtensible Application Markup Language*) e inclui dois espaços *canvas*, um para o esqueleto do usuário e outro para a imagem. Além disso, a interface gráfica apresenta a imagem de uma plataforma Arduino, contendo pinos com suas respectivas numerações ampliadas (para facilitar a visualização de qual saída do Arduino foi ligada ou desligada).

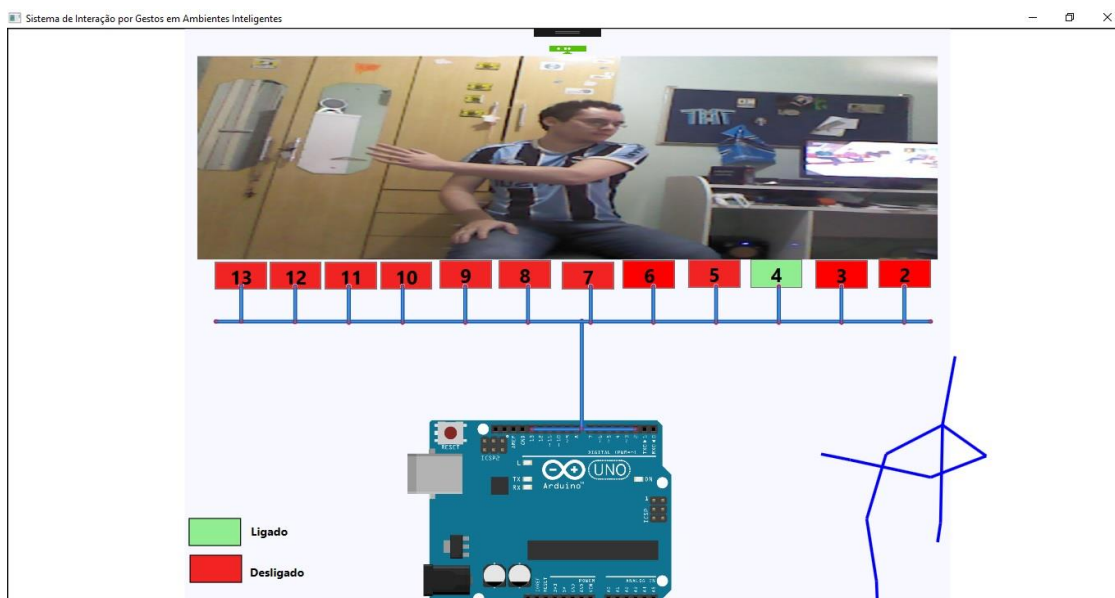


Figura 6 - Interface Gráfica Modo Cadeirante

Na montagem do esqueleto (do usuário cadeirante ou não cadeirante) o fluxo de dados habilita-se quando o sensor é inicializado na classe Kinect através do trecho “Skeleton.Enable()”. Esses quadros de esqueletos são capturados e armazenados em um objeto “frameEsqueleto”. Esses dados do esqueleto são copiados para um *array*, que é percorrido se o esqueleto for detectado.

Para melhor visualização do esqueleto é necessário unir as articulações, através do método “AdicionarLinha”, onde mapeia-se o ponto do esqueleto no espaço de cor, de acordo com a articulação recebida por parâmetro com suas respectivas posições. Por fim desenha-se a linha no *canvas* unindo estas articulações. Definindo-se sua forma como a opção “em pé” ou “cadeirante”, onde na segunda foram excluídas às articulações do quadril e ambas as pernas para melhor visualização na interface Figura 6.

c) Implementação dos gestos

Para a criação de um mecanismo de interpretação gestual é necessária a atualização dos quadros do esqueleto. Dessa forma, o método “AtualizarQuadroAtual”,

no qual o quadro é atualizado constantemente de acordo com o movimento do esqueleto, e dentro deste método chama-se identificação dos gestos. No método “IdentificarGesto”, recebe-se o quadro atual do esqueleto, que contém todas as informações do esqueleto e copia para um *array*, após declaradas as articulações, as quais serão usadas na identificação gestual cujas posições são comparadas para a interpretação do rastreamento do movimento, resultando numa variável do tipo *bool*.

Após a comparação das coordenadas das articulações, o sistema confere se realmente foram rastreadas todas as articulações necessárias. Por exemplo, para realizar o “gestoUm” deve-se levantar a mão direita ou esquerda duas vezes acima da cabeça. Conseqüentemente, as articulações (cabeça, mão esquerda e mão direita) devem estar devidamente rastreadas. A Figura 7 apresenta a implementação do “gestoUm”, que recebe em uma variável do tipo *bool* se a articulação (mão direita ou esquerda) na posição Y é maior que a outra articulação (cabeça) na posição Y, adicionado um contador para o número de vezes que é realizada a ação. O gesto realizado deve ser enviado à classe Arduino para a associação de gesto com o respectivo comando.

```
if (MaoDireitaAcimaCabeça != resultadoMaoDireitaAcimaCabeça)
{
    MaoDireitaAcimaCabeça = resultadoMaoDireitaAcimaCabeça;
    if (MaoDireitaAcimaCabeça)
    {
        ContadorMaoDireitaAcimaCabeça++;
        if (ContadorMaoDireitaAcimaCabeça == 2)
        {
            Gesto = GestoEnum.MaoDireitaAcimaDaCabeça.ToString();
            arduino.AssociarGestoComando(Gesto);
            ContadorMaoDireitaAcimaCabeça = 0;
        }
    }
}
else if (MaoEsquerdaAcimaCabeça != resultadoMaoEsquerdaAcimaCabeça)
{
    MaoEsquerdaAcimaCabeça = resultadoMaoEsquerdaAcimaCabeça;
    if (MaoEsquerdaAcimaCabeça)
    {
        ContadorMaoEsquerdaAcimaCabeça++;
        if (ContadorMaoEsquerdaAcimaCabeça == 2)
        {
            Gesto = GestoEnum.MaoEsquerdaAcimaDaCabeça.ToString();
            arduino.AssociarGestoComando(Gesto);
            ContadorMaoEsquerdaAcimaCabeça = 0;
        }
    }
}
```

Figura 7 – Implementação do “GestoUm”

Da mesma forma, como apresentada a implementação do “gestoUm”, é desenvolvido o “gestoDois”, o qual realiza o movimento da mão direita para o lado esquerdo e também o movimento da mão esquerda para o lado direito. Ao construir parte do “gestoTres”, é utilizado um conceito diferente da implementação dos gestos

anteriores: o cálculo da distância entre dois pontos para simular o movimento de “palmas”, por meio do teorema de Pitágoras ($d = \sqrt{(x_1 - x_2)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$), como é ilustrado na Figura 8.

```
private double CalcularDistanciaArticulacoes(Joint articulacao1, Joint articulacao2)
{
    double distanciaX = articulacao1.Position.X - articulacao2.Position.X;
    double distanciaY = articulacao1.Position.Y - articulacao2.Position.Y;
    double distanciaZ = articulacao1.Position.Z - articulacao2.Position.Z;
    double resultado = Math.Sqrt(Math.Pow(distanciaX, 2) + Math.Pow(distanciaY, 2) + Math.Pow(distanciaZ, 2));
    return resultado;
}
```

Figura 8 – Cálculo da distância entre dois pontos

No movimento de “palmas” recebe-se a distância atual entre os pontos. Caso a distância atual seja menor que 0.1 e a distância anterior menor que 0.1, então o movimento é rastreado corretamente. Com “palmas ativadas”, a aplicação realiza uma chamada ao método que associa o gesto ao comando (que envia uma mensagem ao microcontrolador) acionando o pino definido no *xml*. Para desligar o dispositivo pelo “gestoTres”, é implementado o gesto de aceno de ambas as mãos. Esse trecho segue a mesma linha de implementação, sendo comparadas as posições das articulações, dando resultado ao movimento.

A implementação da associação de gestos a comandos é desenvolvida na classe Arduino. O método “AssociarGestoComando”, que recebe por parâmetro uma variável do tipo *String*, enviada da classe Gerenciador pelo método “IdentificarGestos”. Essa mensagem é concatenada com o pino ou os pinos, definidos para determinado gesto no *xml*, e enviada ao microcontrolador. Na implementação no microcontrolador recebe-se essa mensagem (via conexão serial) convertida do formato *byte* para caractere, sendo concatenados para formar, novamente, a mensagem. Nessa mensagem é identificado o pino e qual ação (ligar ou desligar) ocorrerá no dispositivo pertencente ao ambiente.

4. Cenário de Validação e Testes

Considerando que este trabalho tem como objetivo o desenvolvimento de um sistema, que possibilita a interação do usuário com ambientes inteligentes por meio de gestos, é possível aplicá-lo, além do cenário escolhido, em vários outros. Por exemplo, sistemas automatizados de irrigação, sistemas de automação industrial e sistemas de automação predial/residencial.

Entretanto, neste contexto, optou-se por testar e validar o projeto por meio da simulação de uma casa automatizada (em uma maquete) ilustrada na Figura 9. Essa casa automatizada possui *leds* para iluminação, *cooler* para climatização e um motor para controle do portão frontal.

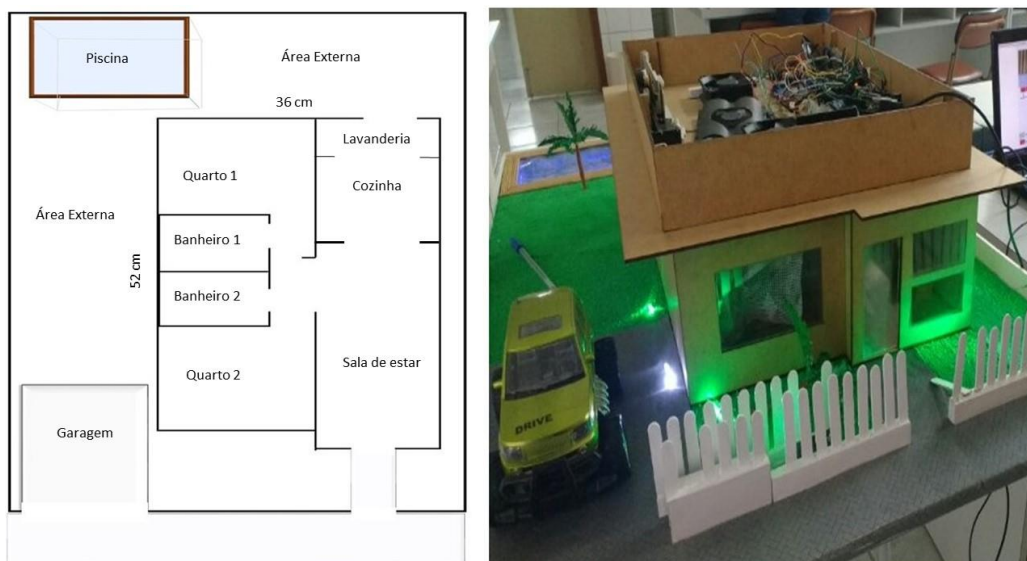


Figura 9 - Planta Maquete / Maquete Casa Automatizada

Considerando o cenário de teste, as configurações dos arquivos de perfil foram ajustadas da seguinte maneira: o modo cadeirante ativo, com uma angulação de “5” graus, pino “2e3” para o “gestoUm”, pino “4” para o “gestoDois” e pino “6” para o “gestoTres”. Com o modo cadeirante ativado, a interface renderizou o esqueleto do usuário cadeirante.

No teste realizado na maquete da casa automatizada optou-se, devido a necessidade para realização da ação, a utilização de dois pinos (pinos dois e três) para o “gestoUm” de abrir e fechar o portão, o pino quatro para realizar o acionamento das lâmpadas externas através do “gestoDois” e, o pino seis para ligar o sistema de ventilação executando o “gestoTres”. A realização do “gestoUm” permitiu a abertura e fechamento do portão (motor de corrente contínua), enquanto que a efetivação do “gestoDois” gerou a ativação das luzes externas (*leds*) da casa e o “gestoTres” concretizou o acionamento ou desacionamento do sistema de ventilação (*cooler*).

5. Conclusão

Este trabalho abordou a proposta de um sistema para reconhecimento de gestos em um ambiente de interação multimodal, em que pode ser utilizada uma outra forma de interação humano-computador em cenários automatizados. Além disso, a partir dos estudos nos trabalhos relacionados foi identificado que uma das principais características é a interação nesses ambientes. Ao desenvolver essa interação humano-computador espera-se conseguir aproximar a naturalidade da comunicação das pessoas com o cenário automatizado, utilizando gestos.

Sendo assim, pode-se verificar a importância da proposta escolhida, por esta tentar oferecer maior conforto e praticidade aos seus usuários e, ainda, auxiliar as pessoas em tarefas cotidianas. Existem diversos ambientes automatizados e, de fato, a tecnologia está presente, implicando diretamente na vida das pessoas, tanto nas tarefas domésticas como nas profissionais.

Do ponto de vista crítico, percebe-se na proposta a limitação na criação e na diversidade dos gestos por usar apenas os membros superiores do esqueleto. Isso, dificulta relacionar o gesto com a ação, principalmente, por ter que apresentar um gesto

diferente para cada função, não podendo usar os membros inferiores e nem repetir os movimentos combinados.

Para trabalhos futuros sugere-se ampliar o número de gestos e, ainda, aproveitar melhor as funcionalidades do que o sensor Kinect proporciona como, por exemplo, um comando de voz associado ao gesto. Ainda, é possível desenvolver um sistema para cadeirantes que não possuem os movimentos dos membros superiores bem articulados, os quais poderiam utilizar (eventualmente) os movimentos com a cabeça e com a face, associados aos comandos de voz.

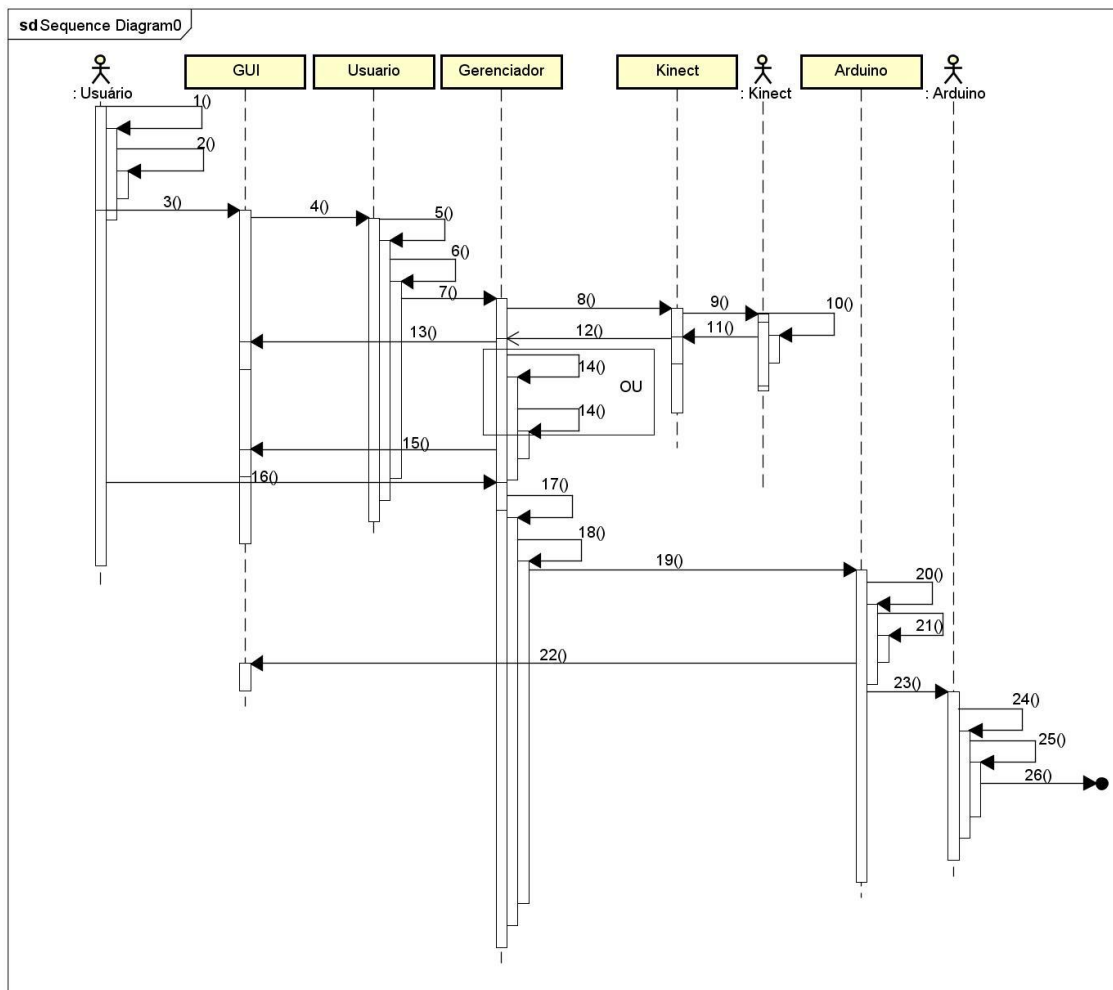
Por fim, a validação da proposta foi realizada no TFG 2 contemplando um cenário capaz de simular uma casa automatizada composta de um sistema de iluminação, ventilação e de abertura e fechamento de portão.

Referências

- Almeida, F. B. (2013), “ Sistema interativo baseado em gestos para a utilização de comandos no computador”, Monografia Graduação, Universidade de Brasília Faculdade do Gama, Distrito Federal.
- Arduino (2016), “Introduction”. Acesso em Março de 2016. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>.
- Augusto J. C., Nakashima H. e Aghajan H. (2009), “Ambient Intelligence and Smart Environments: A State of the Art” Handbook on Ambient Intelligence and Smart Environments, Springer Verlag, New York, Estados Unidos da América.
- Benyon D. (2011), “Interação Humano Computador” 2ª ed. editora Pearson Prentice Hall, São Paulo.
- Bolzani C. A. M. (2004), “Residências Inteligentes: um curso de domótica” 1ª ed. Editora Livraria da Física, São Paulo.
- Cardoso S. G. (2014), “MICROSOFT KINECT: CRIE APLICAÇÕES INTERATIVAS”, Casa do código, São Paulo.
- Carvalho J. O. F. (2003), “O papel da interação humano-computador na inclusão digital”, 15ª ed. Transinformação Campinas, 75-89, São Paulo.
- Campos C. A. T. (2009), “Comportamento das componentes de sistema multi – toque baseados em reflexão interna total confinada”, Dissertação Programa de Pós-graduação em Informática, PUC-Rio, Rio de Janeiro.
- Catuhe D. (2012), “Programming with the Kinect for Windows Software Development Kit ”, Microsoft Press A Division of Microsoft Corporation, Washington, Estados Unidos da América.
- Ferraz A. S. e D. J. S. Messias. (2011), “Computação Ubíqua”, Universidade de Pernambuco, Graduação em Ciência da Computação, Recife, Pernambuco.
- Ghirotti S. E. e Morimoto C. H. (2010), “Um sistema de interação em gestos manuais tridimensionais para ambientes virtuais”, Departamento de Ciência da Computação, IME/USP, São Paulo.

- Guerra C. A. N. (2007) “Um modelo para ambientes inteligentes baseado em serviços web semânticos”, dissertação Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.
- Microsoft (2012), “Kinect for Windows sensor components and specifications ”. Acesso em março de 2016. Disponível em:< <https://msdn.microsoft.com/en-us/library/jj131033.aspx> >.
- Microsoft (2016), “Kinect para XBOX 360”. Acesso em Junho de 2016. Disponível em:<<http://www.xbox.com/pt-BR/xbox360/accessories/kinect/kinectforxbox360>>.
- Moreira C. C. (2013), Dissertação de mestrado, Engenharia Elétrica – Universidade Federal do Pará – PA.
- Muratori J. R. e Dal Bó P. H (2011), “Automação residencial: histórico, definições e conceitos” Revista O setor Elétrico, Capítulo 1. Editora Atitude Editorial, São Paulo.
- Oliveira, A. S. (2014), “ Protótipo de uma interface homem-computador baseada em interação por voz, gestos, multitoque e transparência para automação predial”, tese de doutorado, UFRJ/ COPPE, Rio de Janeiro.
- Peçanha C. C. (2012), “Ambiente Inteligente Controlado por Dispositivos Móveis”, trabalho de conclusão do curso de Ciência da Computação, Universidade Vila Velha, Espírito Santo.
- Penido E. C. C. e Trindade R. S. (2013), “Microcontroladores”, Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Minas Gerais.
- Pistolozzi J. F. O. (2009), “Considerações para implantação de sistemas de automação predial”, Monografia curso de Engenharia de Controle e Automação, Universidade Federal de Ouro Preto, Minas Gerais.
- Silva, B. C. R. e Cândido L. A. A. (2011), “Sistema de controle residencial baseado na plataforma arduino”, Monografia, Instituto Unificado de Ensino Superior Objetivo, IUESO, Goiás.
- Spinola R. O. (2010), “ Diagrama de Classes na Prática – Estudo de Caso ”, Revista Engenharia de Software Magazine, 20ª ed.
- Souza F., Correa B. B. e Machado A. (2013), “Ambientes Inteligentes: Uma aplicação utilizando DOMUS”, Anais do EATI - Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação, Universidade Federal de Santa Maria, Rio Grande do Sul.
- Souza D. J. (2005), “Desbravando o PIC”, 4ª ed. Editora Érica, São Paulo.
- Talarico Neto A. (2011), “Uma abordagem para projeto de aplicações com interação multimodal na Web”, Tese doutorado Universidade de São Paulo, São Paulo.
- Xbox (2016), “Kinect for Xbox 360” Acesso em Maio de 2016. Disponível em:
< <http://www.xbox.com/en-US/xbox-360/accessories/kinect> >.
- Wendling M. (2010), “Sensores” Acesso em Março de 2016. Disponível em:
<<http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>>.

Apêndices



Apêndice 1 - Diagrama de sequência

Apêndice 2 – Identificação da numeração do diagrama de sequência

Número	Método	Descrição
1	SetConfiguracaoUsuario.xml()	Configuração do arquivo perfil do usuário
2	SetConfiguracaoArduíno.xml()	Configuração do arquivo de perfil do Arduíno
3	Window_Loaded_1()	Carrega a interface gráfica
4	ConsultarCadeiranteOuEmPe()	É chamado o método de consulta
5	GetXML()	Busca o arquivo de perfil xml
6	ConfiguraCadeiranteOuEmPe()	Identifica e armazena os dados recebidos para configuração do sistema
7	InicializarKinect(int valorAngulacao, string categoria)	Envia a angulação e a categoria para o gerenciador

8	InicializarSensorKinect(int anguloElevacaoInicial)	Gerenciador chama método da classe Kinect
9	InicializaSensor(angulo)	Classe Kinect comunica ao sensor Kinect
10	InicializaComponentes()	O sensor Kinect inicializa seus componentes
11	EnviaDados(KinectSensor kinect)	Envia dados para a classe Kinect
12	kinect = InicializarKinect	Gerenciador recebe os dados do sensor
13	MontarImagemColorida ()	Monta a imagem colorida na interface gráfica
14	MontarEsqueletoCadeirante () OU MontarEsqueletoEmPe ()	A interface gráfica monta esqueleto cadeirante ou em pé
15	AtualizarCanvasEsqueleto()	Atualiza o esqueleto na interface gráfica
16	UsuárioRealizaGesto()	O usuário realiza o gesto
17	AtualizarQuadroAtual(object sender, SkeletonFrameReadyEventArgs e)	O quadro atual do esqueleto é atualizado
18	IdentificarGesto(quadroAtual)	Identifica o gesto realizado
19	AssociarGestoComando(Gesto)	Associa o gesto ao determinado comando contendo o pino configurado no arquivo de perfil do Arduino
20	GetXML()	Busca arquivo de perfil do Arduino
21	ConsultarConfiguracaoArduino()	Identifica e armazena os dados “pinoGesto”
22	MudaCorBotaoPortaLigadoOuDesligado()	Muda a cor do botão na interface gráfica
23	EnviaComandoArduino(Comando)	Envio do comando ao Arduino
24	LerStringViaComunicacaoSerial()	Converte e lê a palavra recebida
25	IdentificaStringPino()	Identifica pino de acordo com a palavra
26	AcionaOuDesacionaDispositivosAutomacao()	Liga ou desliga os dispositivos