

UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO
CARLOS
DEPTO. DE ENGENHARIA ELÉTRICA E
DE COMPUTAÇÃO

**Construção de um Assistente Pessoal
Open Source**

Autor: Guilherme Cabral da Silva, nº. USP: 9403343
Autor: Samuel S. do Espírito Santo, nº. USP: 9393221
Orientador: Prof. Dr. José Roberto B. A. Monteiro

Guilherme Cabral da Silva,
Samuel Santos do Espírito Santo

Construção de um Assistente Pessoal Open Source

Trabalho de conclusão de curso de Engenharia Elétrica apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo - USP, sob a orientação do Prof. Dr. José Roberto.

Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

Orientador: Prof. Dr. José Roberto B. A. Monteiro

São Carlos
05 de Novembro de 2019

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

S586c

Silva, Guilherme Cabral da
Construção de um assistente pessoal open source / Guilherme
Cabral da Silva, Samuel Santos do Espírito Santo; orientador
José Roberto B. A. Monteiro. -- São Carlos, 2019.

Monografia (Graduação em Engenharia Elétrica com ênfase em
Eletrônica) -- Escola de Engenharia de São Carlos da
Universidade de São Paulo, 2019.

1. Código aberto. 2. Desenvolvimento em comunidade.
3. Raspberry. 4. Assistente pessoal. 5. Python. I. Espírito
Santo, Samuel Santos do. II. Título.

FOLHA DE APROVAÇÃO

Nome: Guilherme Cabral da Silva

Título: "Construção de um Assistente Pessoal Open Source"

Trabalho de Conclusão de Curso defendido e aprovado
em 25/11/2019,

com NOTA 8,7 (oito, sete), pela Comissão Julgadora:

*Prof. Associado José Roberto Boffino de Almeida Monteiro -
Orientador - SEL/EESC/USP*

Prof. Dr. Maximiliam Luppe - SEL/EESC/USP

*Prof. Associado Evandro Luis Linhari Rodrigues - SEL/EESC/USP
(docente aposentado)*

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

FOLHA DE APROVAÇÃO

Nome: Samuel Santos do Espírito Santo

Título: "Construção de um Assistente Pessoal Open Source"

Trabalho de Conclusão de Curso defendido e aprovado
em 25/11/2019,

com NOTA 8,7 (oito, sete), pela Comissão Julgadora:

*Prof. Associado José Roberto Boffino de Almeida Monteiro -
Orientador - SEL/EESC/USP*

Prof. Dr. Maximiliam Luppe - SEL/EESC/USP

*Prof. Associado Evandro Luis Linhari Rodrigues - SEL/EESC/USP
(docente aposentado)*

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Agradecimentos

A Deus, em primeiro lugar, que nos proporcionou essa oportunidade e nos capacitou e conduziu ao longo dessa jornada.

Aos nossos pais e irmãos, por todo o apoio, motivação e por sempre proporcionarem as melhores condições para alcançarmos os nossos objetivos.

Ao nosso professor Dr. José Roberto B. A. Monteiro pela orientação, apoio, paciência e oportunidade de realizar esse projeto.

E aos nossos amigos que nos acompanharam durante toda essa trajetória, nos auxiliando em diversos momentos.

Resumo

Este projeto busca desenvolver um assistente pessoal de baixo custo, que permita a democratização do conhecimento, da tecnologia e que possui flexibilidade para o desenvolvimento de novas funcionalidades. Tendo como principal elemento uma Raspberry Pi 3, o assistente pessoal procura servir como base para o desenvolvimento de projetos futuros e disseminar os conhecimentos em IoT e assistentes pessoais. Para tal este conta com algumas funções atrativas, como, agenda, sistema de segurança, verificação da temperatura e umidade ambiente e interação por comando de voz. Este trabalho tem ainda como um dos seus principais objetivos a disponibilização do seu código de forma aberta (open-source), e visa o auxílio da comunidade para desenvolvimentos futuros (Community development).

Palavras Chaves: Código aberto, Desenvolvimento em Comunidade, Raspberry, Assistente Pessoal, Python.

Abstract

This project seeks to develop a low-cost personal assistant that allows the democratization of knowledge, technology and has the flexibility to develop new features. With a Raspberry Pi 3 as its main element, the personal assistant looks to serve as a base project for future developments and to disseminate the knowledge of IoT and Personal Assistants. With that in mind it counts with some attractive functions such as schedule, security system, environment temperature and humidity checking and voice command interaction. This work also has as one of its main objectives the availability of its open source code, and aims to help the community for future development.

Keywords: Open Source, Community Development, Raspberry, Personal Assistant, Python.

Lista de ilustrações

Figura 1 – Raspberry Pi 3 Model B	21
Figura 2 – Display LCD 3.5inch	22
Figura 3 – Sensor de presença PIR HC-SR501	24
Figura 4 – Hardware PIR	25
Figura 5 – Ajustes de operação do PIR	25
Figura 6 – Sensor de temperatura DHT11	26
Figura 7 – WebCam Goldship 3817	27
Figura 8 – Super Mini USB 2.0 Microphone	28
Figura 9 – Servo motor TowerPro MG995	29
Figura 10 – PCA9685 Adafruit	30
Figura 11 – Disposição do sistema operacional no computador	31
Figura 12 – Estrutura HTML	34
Figura 13 – Estrutura HTML (1)	35
Figura 14 – Estrutura HTML (2)	35
Figura 15 – Estrutura HTML (3)	35
Figura 16 – Estrutura HTML com componente de javascript	36
Figura 17 – Arquitetura do Sistema	43
Figura 18 – Cartão micro SD	44
Figura 19 – Configurações no MAC	45
Figura 20 – Terminal Putty	46
Figura 21 – Interface DreamWeaver	48
Figura 22 – Função Ajax para o Controle dos servos Motores	49
Figura 23 – Código HTML, CSS - Temperatura e Umidade	49
Figura 24 – Código Python - Temperatura e Umidade	50
Figura 25 – Código HTML – complete, Incomplete	51
Figura 26 – Método de Tarefa Completada	51
Figura 27 – Exemplo arquivo Python	52
Figura 28 – Exemplo arquivo kivy	53
Figura 29 – Tela kivy	54
Figura 30 – Exemplo código Python com kivy	55
Figura 31 – Interface Kivy (1)	56
Figura 32 – Código Kivy - Interface (1)	56
Figura 33 – Código Kivy - Interface (2)	57
Figura 34 – Kivy - Interface (2)	58
Figura 35 – Código Kivy - Interface (3)	59
Figura 36 – Kivy - Interface (3)	60

Figura 37 – Esquemático motores e RPi	61
Figura 38 – Rotas de controle do motor	62
Figura 39 – Método <i>servo_position</i>	62
Figura 40 – Esquemático PIR	63
Figura 41 – Método de configuração do sensor PIR	64
Figura 42 – Método de envio do email	64
Figura 43 – Esquemático DHT11	65
Figura 44 – Método <i>readDht11</i>	65
Figura 45 – Rota para receber os valores do sensor DHT11	66
Figura 46 – Menu de Configuração RPi, Opções Avançadas	66
Figura 47 – Menu de Configuração RPi, Expandir Sistema	67
Figura 48 – Método teste OpenCV	71
Figura 49 – Método <i>runFaceDetector</i>	71
Figura 50 – Método <i>getImage</i>	72
Figura 51 – Método <i>sendVideo</i>	72
Figura 52 – Método <i>generateVideo</i>	73
Figura 53 – Endereço <i>cameraview</i>	73
Figura 54 – Função <i>create_collection</i>	75
Figura 55 – Função <i>index_face</i>	75
Figura 56 – Função <i>faceMatch</i>	76
Figura 57 – Configuração do arquivo config.txt	77
Figura 58 – Configuração do arquivo <i>asounf.conf</i>	77
Figura 59 – Método <i>speak</i>	79
Figura 60 – Desenvolvimento de skills	79
Figura 61 – Banco de dados python (1)	81
Figura 62 – Banco de dados python (2)	81
Figura 63 – Página de Entrada WEB	84
Figura 64 – Página de Aplicativos	85
Figura 65 – Exemplo Interface Modo Vigia	86
Figura 66 – Interface de Temperatura e Umidade	87
Figura 67 – Interface Agenda	88
Figura 68 – Teste da Variação do Tamanho da Imagem Enviada	90
Figura 69 – Picos de tempo no início do teste	91
Figura 70 – Teste com variação da imagem de entrada	92
Figura 71 – Teste em tempo real	93
Figura 72 – Tempo de Requisição do Controle de Voz	94
Figura 73 – Erro no Reconhecimento de Voz	95
Figura 74 – Teste de temperatura	96
Figura 75 – Teste de Consumo da CPU	97

Lista de tabelas

Tabela 1 – Características técnicas Display LCD 3.5inch	22
Tabela 2 – Características Display LCD	23
Tabela 3 – Características técnicas PIR	24
Tabela 4 – Características técnicas DHT11	26
Tabela 5 – Características técnicas WebCam Goldship 3817	27
Tabela 6 – Características técnicas Super Mini USB 2.0 Microphone	28
Tabela 7 – Características técnicas TowerPro MG995	28

Sumário

Lista de ilustrações	6	
Lista de tabelas	8	
Sumário	9	
1	INTRODUÇÃO	13
1.1	Objetivo	14
1.2	Organização do Trabalho	15
2	FUNDAMENTOS	16
2.1	Produto Final	16
2.2	Sistemas Embarcados	16
2.3	IoT	16
2.4	Sensores	17
2.5	Atuadores	17
2.6	Computação na Nuvem	18
2.7	Open Source	18
3	MATERIAIS	20
3.1	Estrutura de Hardware	20
3.1.1	Raspberry Pi 3 Model B	20
3.1.2	LCD	21
3.1.3	Sensores	23
3.1.3.1	Sensor de Presença	23
3.1.3.2	Sensor de Temperatura e Umidade	25
3.1.3.3	Câmera	26
3.1.4	Mini Microfone USB	27
3.1.5	Servo Motor	28
3.2	Sistemas Operacionais	30
3.2.1	Raspbian	31
3.3	Estrutura de Software	32
3.3.1	Python	32
3.3.2	Web Server	32
3.3.2.1	Flask	33
3.3.3	Páginas WEB	33
3.3.3.1	HTML	33

3.3.3.2	CSS	34
3.3.3.3	JavaScript	35
3.3.4	Banco de Dados	36
3.3.4.1	SQLite	37
3.3.5	Interface Homem-Máquina	38
3.3.5.1	Interface Gráfica	38
3.3.5.2	Visão Computacional	38
3.3.5.2.1	OpenCV	39
3.3.5.3	Amazon Web Services	40
3.3.5.3.1	AWS Rekognition	40
3.3.5.4	Comando de Voz	41
3.3.5.4.1	Google Assistant	42
4	DESENVOLVIMENTO	43
4.1	Arquitetura do Sistema	43
4.2	Configuração da Raspberry PI	44
4.3	Configuração do Servidor Flask	47
4.4	Web	47
4.4.1	DreamWeaver	48
4.4.2	Página Inicial e de Aplicativos	48
4.4.3	Página Modo Vigia	49
4.4.4	Página Sensoriamento (Temperatura e Umidade)	49
4.4.5	Página Agenda	50
4.5	Interface Gráfica	51
4.5.0.1	Geração de Telas	55
4.6	Sistema de Segurança	60
4.6.1	Configuração do Motor	60
4.6.2	Configuração do Sensor PIR	62
4.6.3	Configuração DHT11	64
4.6.4	Configuração OpenCV	66
4.6.4.1	Preparando Ambiente OpenCV	70
4.6.4.2	Implementação Geral	71
4.6.4.3	Implementação WEB	72
4.6.5	Configuração AWS Rekognition	73
4.7	Comando de Voz	76
4.7.1	Configuração do Google Assistant	76
4.7.2	Desenvolvimento de Skills	78
4.8	Desenvolvimento do Banco de Dados	80
4.9	Característica Open-Source	81

5	RESULTADOS E DISCUSSÕES	83
5.1	Interação com o Usuário	83
5.1.1	Web	83
5.1.1.1	Página Inicial	83
5.1.1.2	Página de Aplicativos	84
5.1.1.3	Página Modo Vigia	85
5.1.1.4	Sensor de Temperatura e Umidade	86
5.1.1.5	Agenda WEB	87
5.2	Sistema de Segurança	88
5.2.1	Controle Câmera	88
5.2.2	Detector de Presença	89
5.2.3	Reconhecimento Facial	89
5.2.3.1	Testes AWS Rekognition	89
5.3	Controle de Voz	93
5.4	Performance do Sistema	95
6	CONCLUSÃO	98
6.1	Aprimoramentos Futuros	99
	Referências	100
	APÊNDICES	106
	APÊNDICE A – GERENCIADOR DO SERVIDOR: <i>SERVER.PY</i>	107
	APÊNDICE B – PROGRAMA <i>MAIN.PY</i>	110
	APÊNDICE C – PROGRAMA <i>AWS CONNECTION.PY</i>	112
	APÊNDICE D – PROGRAMA <i>CAMERA.PY</i>	113
	APÊNDICE E – PROGRAMA <i>KIVY GUI.PY</i>	115
	APÊNDICE F – PROGRAMA <i>PERSONAVERIFICATION.PY</i>	119
	APÊNDICE G – PROGRAMA <i>SECURITY.PY</i>	121
	APÊNDICE H – PROGRAMA <i>SENSOR.PY</i>	122
	APÊNDICE I – PROGRAMA <i>SERVOMOTOR.PY</i>	123
	APÊNDICE J – CÓDIGO PÁGINA PRINCIPAL HTML	124

APÊNDICE K – CÓDIGO HTML PÁGINA APLICATIVOS	127
APÊNDICE L – CONTROLE MOTORES E CAMERA HTML	130
APÊNDICE M – CÓDIGO HTML TEMPERATURA E UMIDADE	133
APÊNDICE N – TODO HTML	138

1 Introdução

A humanidade, com o passar dos anos, vem encontrando novas formas cada vez mais orgânicas de se utilizar da tecnologia no auxilio das tarefas diárias. Atualmente uma das áreas de maior crescimento dentro do ramo tecnológico são os assistentes pessoais e os IoTs[1][2]. Apesar do grande foco dado atualmente aos assistentes pessoais, a sua concepção é antiga. Desde 1987 a Apple vem tentando implantar tecnologias que buscam realizar tais funções, como o seu “*Knowledge navigator*” [3]. Os assistentes pessoais têm como principal objetivo simplificar ações diárias que vão desde grandes tarefas como controlar uma frota de carros a até fazer as compras do mês [4] [5].

Os elementos IoTs, ou *Internet of Things*, por sua vez são:

“Elementos que tem como principal função fazer com que objetos (moveis, imoveis e dispositivos) inteligentes sejam o bloco de desenvolvimento do espaço físico-cibernético permitindo que os mesmos interajam entre si e com os usuários realizando ações de forma a facilitar a vida dos seus usuários.” [6][7]

Porém apesar de todo o esforço de grandes empresas em criar tecnologias como essas, poucas acabam sendo acessíveis ao público em geral. Principalmente na sociedade brasileira onde essas tecnologias apresentam preços elevados e até mesmo um acesso restrito.

Apesar do acesso restrito no Brasil em outros lugares do mundo cada vez mais assistentes pessoais vem surgindo como as opções Alexa (desenvolvida pela Amazon), o Google Home (Google), a Cortana (Microsoft), dentre outros. Essas tecnologias por sua vez buscam integrar a modernização ao dia a dia das pessoas de forma estruturada e eficiente.

Dessa forma as pessoas podem se sentir cada vez mais confortáveis com os equipamentos IoTs e com o auxilio da tecnologia. Motivado por essa nova tendência que nasce a ideia do assistente pessoal apresentado neste trabalho o qual recebeu o nome de MUSK, o qual recebeu esse nome como uma homenagem ao cientista e empreendedor Elon Musk, que vem tendo importantes contribuições no setor tecnológico. Um assistente pessoal de código aberto que permite acesso a tecnologias de ponta, tornando dessa forma o conhecimento deste tipo de aplicação acessível.

1.1 Objetivo

Este trabalho tem como objetivo a idealização e desenvolvimento de um assistente pessoal capaz de realizar tarefas diárias a partir do controle de dispositivos externos. Foi então construído um protótipo para que dessa forma fosse possível verificar o funcionamento do assistente pessoal aqui projetado. Por sua vez, o protótipo é uma iniciativa que serve como base para o desenvolvimento de projetos mais complexos e diversificados, como os assistentes pessoais comerciais (e.g.: Google Home, Echo Dot e etc).

Assim o MUSK foca na habilidade de ser integrado a diversos dispositivos, módulos e atuadores, garantindo a sua flexibilidade e adaptação a necessidade do usuário e do próprio projetista.

Em relação à iniciativa Open Source, o protótipo permite que pessoas de diversas classes sociais possam ter acesso a ferramentas de desenvolvimento de tecnologias avançadas e em alto crescimento dentro do mercado atual.

O grande diferencial do MUSK apresenta-se no fato deste ser um dos primeiros projetos de assistente pessoal de código aberto e que é capaz de contar com o auxílio de construção por comunidade.

É com o intuito de democratizar a tecnologia permitindo que tanto entusiastas quanto amadores tenham acesso a esta tecnologia em expansão que reside o principal objetivo do MUSK. Para conseguir alcançar tais objetivos utiliza-se elementos consideravelmente mais simples e baratos e tem como intuito contar com um código aberto e com apoio a comunidade.

Uma vez que um dos fundamentos do projeto está em utilizar-se de tecnologia hodiernas o protótipo do MUSK tem como base a plataforma de processamento Raspberry Pi, além de integrar sensores, dispositivos de áudio e imagem, atuadores eletromecânicos. Em relação ao seu software embarcado determinou-se a utilização de um *backend* em Python com o *framework* Flask, o banco de dados em SQLite e um *frontend* em HTML, CSS e Javascript. O projeto ainda integra serviços da Amazon Web Services (AWS) e do Google Cloud.

Por fim pode-se resumir que a grande relevância e objetivo deste projeto está em possibilitar o conhecimento por parte dos alunos de tecnologias que são tendências no mercado atual. Além de permitir o acesso a qualquer pessoa o poder de programar os seus próprios aplicativos e ter o seu assistente pessoal personalizado, contribuindo assim com a área de Internet das Coisas.

1.2 Organização do Trabalho

Este trabalho é dividido em 5 capítulos, contando com a introdução, conforme descrito a baixo:

- **Capítulo 2:** Expõe os fundamentos básicos utilizados no projeto.
- **Capítulo 3:** Apresenta os materiais e ferramentas aplicadas no trabalho, além de detalhar os procedimentos empregados.
- **Capítulo 4:** Discorre sobre os resultados obtidos até o fim do prazo útil.
- **Capítulo 5:** Conclui a discussão sobre o projeto, avaliando seus aspectos positivos e negativos.
- **Apêndices:** Apresenta os códigos elaborados ao decorrer do trabalho.

2 Fundamentos

2.1 Produto Final

Uma vez que o objetivo final deste trabalho é a elaboração de um assistente pessoal e este por sua vez constitui-se como um sistema embarcado, torna-se necessário que se compreenda primeiramente o que é um sistema embarcado. Para tal será estudado suas principais características, seus objetivos e a situação do atual deste sistema no mercado.

2.2 Sistemas Embarcados

Segundo a definição de Stallings[8], um sistema embarcado consiste em um sistema micro-controlado no qual o computador é dedicado para uma aplicação específica, diferentemente de computadores generalizados. Assim, o sistema embarcado realiza um conjunto de tarefas pré-definidas em seu projeto, utilizando os mesmos componentes de um computador em menor quantidade e embutidos em um único chip [8].

Devido a sua versatilidade e acessibilidade, os sistemas embarcados estão cada vez mais difundidos em diversas áreas do cotidiano, como por exemplo [8]:

- Sistemas Automotivos: controle geral do carro, computadores de bordo e telemetria;
- Eletrodomésticos em geral;
- Drivers e periféricos de computador;
- Indústria: automação de processos e gerenciamento do sistema;
- Aparelhos de comunicação: celulares, elementos de redes e tablets.

Assim, a partir deste sistema, é possível a otimização do projeto ao reduzir tamanho, ferramentas computacionais e custos sem que ocorra uma perda de desempenho.

2.3 IoT

O termo Internet das Coisas (IoT) foi criado por Kevin Ashton, e é definida como uma infraestrutura global para a sociedade conectada, possibilitando serviços avançados através de interconexões de coisas (Físicas e Virtuais) baseadas em tecnologias de informação com interoperabilidade existentes ou em desenvolvimento [9].

Atualmente, as grandes empresas de Tecnologia da Informação (TI), como AWS, Google e Microsoft, estão investindo massivamente em IoT, através do desenvolvimento de plataformas em nuvem, com ferramentas que podem ser utilizadas na elaboração de soluções em internet das coisas.

As áreas de maior impacto pela Internet das Coisas, são [10]:

- Otimização da Produção;
- Gestão de cadeia de fornecimento;
- Acompanhamento e gerenciamento de ativos;
- Tomada de decisões financeiras;
- Experiência do cliente.

2.4 Sensores

Um sensor é definido como um componente eletrônico que tem a capacidade de transformar um evento físico em informação elétrica [11]. A partir desse dado, o sistema de controle responsável pela leitura do dispositivo executa funções definidas anteriormente em sua programação. No entanto, existem dois tipos de sinais emitidos por sensores: analógicos e digitais.

O sensor analógico pode assumir quaisquer valores de tensão ao longo do tempo, desde que esteja dentro de sua faixa de operação. Assim, antes de executar os comandos programados no processador é necessário tratar o sinal analógico, ou seja, é realizada uma conversão AD [12].

O sensor digital trabalha com valores discretos. A partir de circuitos eletrônicos, o sinal destes sensores são convertidos resultando em apenas dois valores de operação ao longo do tempo. Estes valores podem ser interpretados de acordo com a escala de bits definida [12].

2.5 Atuadores

Atuadores são por definição [13] elementos que convertem energia elétrica, hidráulica ou pneumática em mecânica e são utilizados nesse projeto para movimentar a câmera, caso isto seja solicitado pelo usuário via integração Web. É possível classificar-se os atuadores em dois tipos:

- **Atuadores a base de Fluidos:** São os atuadores que utilizam-se de ar ou óleo para gerar o movimento. Estes possuem algumas características particulares como, o seu

controle que normalmente se dá em forma de estados (ora contraído, ora acionado) e não possuem controle quanto a velocidade de atuação.

- **Atuadores Eletromagnéticos:** Estes utilizam-se o fenômeno de que a energia elétrica possui, uma vez que quando a mesma é capaz de gerar um campo magnético ao ser transmitida através de um elemento condutor. Tais elementos são conhecidos como motores elétricos e por sua vez, apesar de serem mais caros que os atuadores hidráulicos e pneumáticos eles possuem um controle bem mais elevado.

2.6 Computação na Nuvem

Computação na nuvem, ou Cloud Computing como é mais popularmente conhecido, é a oferta de poder computacional, armazenamento de dados, aplicativos e outros serviços [14]. Devido a quebra de paradigma do oferecimento do poder computacional como um serviço e não mas como um produto abrem-se espaços para que componentes de hardware mais modesto sejam capazes de realizar operações mais complexas.

Dentre alguns dos serviços oferecidos estão armazenamento de banco de dados, armazenamento de sites, análise de dados, reconhecimento facial, assistente de voz, entre outros.

Alguns dos serviços populares usados ao longo deste trabalho são:

- AWS
- Google Assistant
- gTTS (Google Text To Speech)

2.7 Open Source

Open source é um termo em inglês que traduzido significa código aberto, ou seja, quando um software é *open source* este tem como finalidade a liberação do código fonte, sem licenças de uso, para a comunidade de desenvolvedores, assim qualquer pessoa pode visualizar o código e sugerir alterações. A *Open Source Initiative* (OSI) é um organização responsável pelo disseminação dessa cultura de software aberto ou livre. A OSI ainda impõe dez condições para um software ser considerado *open source* [15]:

- Distribuição livre;
- O software deve ser distribuído com seu código fonte;

- A licença do software deve fornecer permissões para que modificações sejam realizadas;
- Integridade do autor do código fonte;
- Não discriminação contra pessoas ou grupos;
- Não discriminação contra áreas de atuação do usuário que deseja utilizar o programa;
- Distribuição da licença;
- O programa não pode fazer parte de outro software, sendo que para sua utilização é necessária a distribuição de todo o código;
- O programa não pode colocar restrições em outros programas que estão sendo distribuídos juntos;
- A licença deve permitir que sejam adotadas interfaces, estilos e tecnologias sem restrições.

3 Materiais

3.1 Estrutura de Hardware

O assistente pessoal utiliza de alguns componentes de hardware para a sua construção física. Dentre esses elementos são utilizados: microcomputador, sensores, atuadores e dispositivos de áudio e som. Essa sessão tem como objetivo apresentar essas estruturas com suas especificações técnicas e modos de utilização.

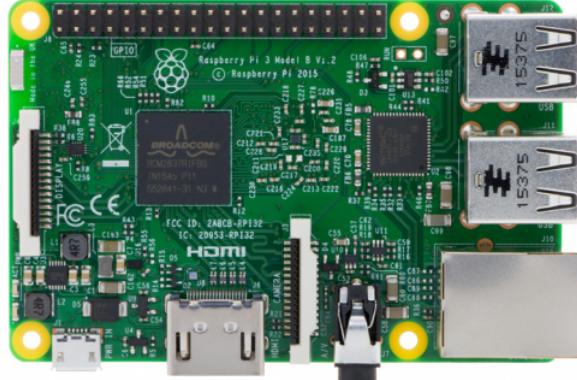
3.1.1 Raspberry Pi 3 Model B

Raspberry Pi (RPi) é um computador completo projetado em apenas uma placa de circuito impresso, desenvolvido pela Fundação Raspberry Pi. Apesar de não possuir um desempenho comparável com a de um computador de mesa moderno, a RPi é um computador com grande capacidade para aquilo que se propõe, além de requisitar um baixo consumo de energia.

Neste projeto foi utilizado a RPi 3 Model B, ilustrado na Figura 1, que possui um System On Chip Broadcom BCM2837, cujo processador é um ARMv8 Cortex-A53 QuadCore com um *clock* de 1.2GHz. Sua memória RAM é de 1GB que por sua vez é mais do que suficiente para a instalação de um Sistema Operacional e um Web Server sem perda em sua capacidade de execução. Além disso a RPi ainda possui: [16]

- Bluetooth 4.1 BLE integrado;
- Adaptador Wifi 802.11n integrado;
- GPIO de 40 pinos;
- Interface para câmera e display;
- Slot para cartão SD;
- Conector de vídeo HDMI;
- 4 portas USB;

Figura 1 – Raspberry Pi 3 Model B



Fonte:<<https://br.rsdelivers.com/product/raspberry-pi/raspberry-pi-b/raspberry-pi-model-b-sbc-computer-board/8111284>>

A versão três da RPi é comparada, por muitos, a um PC devido ao seu elevado desempenho [17]. Esse avanço de *hardware* em relação as versões anteriores, deve-se a utilização de um processador de arquitetura *Advanced RISC Machine* (ARM). Essa arquitetura é conhecida pela sua simplicidade, desempenho, baixo custo e consumo. Suas principais características são [18]:

- Registradores em 32 bits;
- Dois conjuntos de instruções: ARM (32 bits) e THUMB (16 bits);
- Permite a execução condicional de várias instruções;
- Formato de instruções de 3 endereços: 2 registradores operandos e 1 registrador resultado, independentemente especificados;
- Uniformidade e tamanhos fixos dos campos das instruções para facilitar a decodificação;
- Manipulação dos periféricos de I/O como dispositivos mapeados em memória;
- Tamanho do núcleo reduzido.

3.1.2 LCD

Para que fosse possível a interação visual com o usuário instalou-se uma tela LCD sensível ao toque [19][20]. A tela LCD conta com 26 pinos, permitindo assim a conexão de outros sensores e atuadores. As características da tela LCD são apresentadas na Tabela 1.

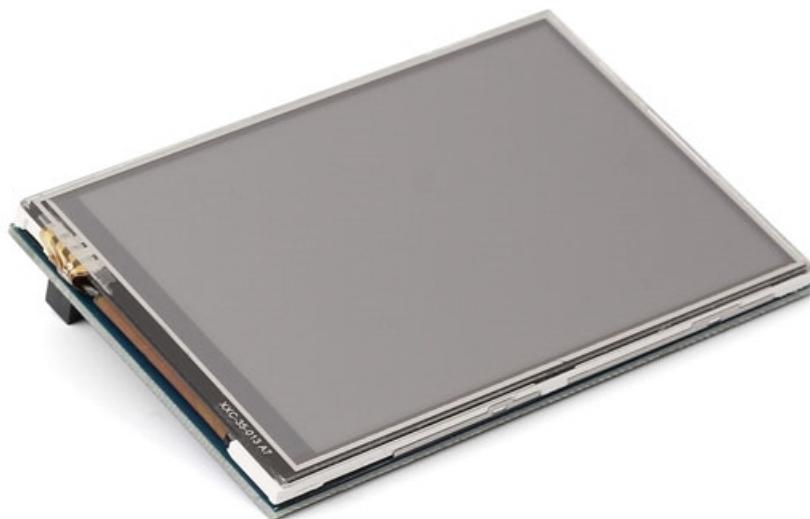
Tabela 1 – Características técnicas Display LCD 3.5inch

Variável	Range
SKU	MPI3501
Tipo de LCD	TFT
Interface LCD	SPI (Fmax: 32MHz)
Tipo de Touch-Screen	Resistivo
Controlador de Touch-Screen	XPT2046
Resolução	320 x 480 (Pixel)
Quantidade de Cores	65536
Corrente de Backlight	120mA
Dissipação de Energia	0.13A*5V ou 0,65W
Temperatura de Trabalho	-20 60 °C
Dimensões da Tela	85,42 x 55,60 (mm)
Dimensões Combinadas	118 x 72 x 34 (mm)
Peso	75 g

Fonte:<<https://www.filipeflop.com/produto/display-lcd-tft-touch-3-5-raspberry-pi/>>

A tela instalada está presente na Figura 2.

Figura 2 – Display LCD 3.5inch



Fonte:<<https://www.filipeflop.com/produto/display-lcd-tft-touch-3-5-raspberry-pi/>>

Por fim os pinos de saída estão presentes na Tabela 2.

Tabela 2 – Características Display LCD

PIN NO.	Simbologia	Descrição
1, 17	3.3V	Alimentação (3.3V)
2, 4	5.0V	Alimentação (5.0V)
3, 5, 7, 8, 10, 12, 13, 15, 16	NC	Não Conectado
6, 9, 14, 20, 25	GND	Ground
11	TP_IRQ	<i>Touch Panel interrupt-</i> Nível Baixo quando detecta toque
18	LCD_RS	Instruction/Data Register selection
19	LCD_SI / TP_SI	SPI data input of LCD/Touch Panel
21	TP_SO	SPI data output of Touch Panel
22	RST	Reset
23	LCD_SCK / TP_SCK	SPI clock of LCD/-Touch Panel
24	LCD_CS	LCD chip selection, Ativo em nível baixo
26	TP_CS	Touch Panel chip selection, Ativo em nível baixo

Fonte:<http://www.lcdwiki.com/3.5inch_RPi_Display>

3.1.3 Sensores

O MUSK utiliza sensores para detectar e responder estímulos do ambiente em que está posicionado, como: detecção de presença e variações de temperatura.

3.1.3.1 Sensor de Presença

O sensor selecionado, para ser responsável pela verificação de eventuais presenças no ambiente monitorado, é o PIR HC-SR501, ilustrado na Figura 3, uma vez que ele é um dos mais utilizados devido a sua qualidade e facilidade de aplicação. A seguir estão representadas na Tabela 3 as características do sensor [21].

Tabela 3 – Características técnicas PIR

Variável	Range
Tensão de Operação	4,5-20V
Tensão Dados	0V e 3,3V
Distância detectável	3 - 7m
Temperatura de Trabalho	-20 +80°C
Dimensões	LCD _{RS}
Peso	7g

Fonte:<<https://www.robocore.net/loja/sensores/sensor-de-presenca-pir-hc-sr501>>

Figura 3 – Sensor de presença PIR HC-SR501



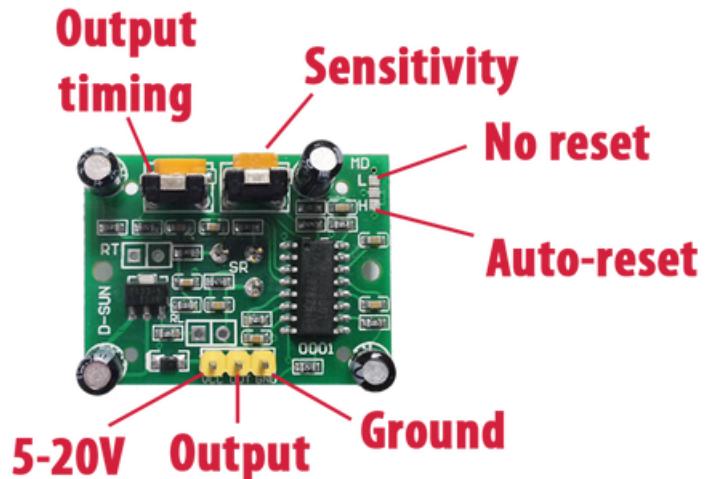
Fonte:<<https://www.marinostore.com/sensores/sensor-de-movimento-e-presenca-pir>>

O PIR possui dois modos de funcionamento: 'Auto Reset' e 'No Reset'.

- **Modo H (Auto Reset):** após uma detecção o sensor reseta o 'tempo de saída' e retorna a função de verificação de movimento após um intervalo de tempo.
- **Modo L (No Reset):** não é necessário que o sensor efetue o reset após a detecção de um movimento.

A configuração do hardware do sensor, é apresentado na Figura 4

Figura 4 – Hardware PIR



Fonte:<<https://fritzing.org/projects/hc-sr501-passive-infrared-sensor-pir>>

O sensor ainda possui um controle de 'sensibilidade' (Sensitivity) e 'tempo de saída' (Output timing), conforme visto na Figura 5.

Figura 5 – Ajustes de operação do PIR



Fonte:<<http://famillemoreau.hopto.org/Arduino/HC-SR501>>

Em relação ao alcance, o PIR possui um campo de atuação razoável. Ele detecta movimentos no formato de um cone de 110 graus a uma distância de 3 a 7 metros.

3.1.3.2 Sensor de Temperatura e Umidade

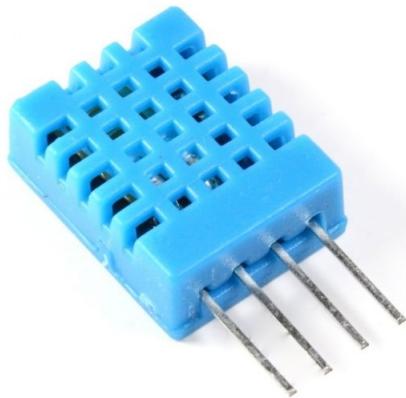
No projeto é utilizado o sensor DHT11, presente na Figura 6, para a aferição da temperatura e umidade ambiente onde se encontra o assistente. O sensor possui quatro pinos e utiliza apenas um para enviar os dados para a RPi. Os demais pinos são utilizados para alimentação e aterramento e o terceiro não é utilizado. As características básicas do sensor estão na Tabela 4 a seguir [22].

Tabela 4 – Características técnicas DHT11

Variável	Range
Alimentação	3 à 5,5 V
Faixa de leitura – Umidade	20 à 80%
Precisão umidade	5%
Faixa de leitura – Temperatura	0 – 50 °C
Precisão temperatura	+/- 2 °C

Fonte: Autoria Própria

Figura 6 – Sensor de temperatura DHT11

Fonte:<<https://www.filipeflop.com/produto/sensor-de-umidade-e-temperatura-dht11/>>

3.1.3.3 Câmera

A câmera utilizada para a obtenção e *streaming* de imagens foi a câmera modelo Goldship 3817, representada na Figura 7. A qual conta com uma resolução de 1.3M pixels e foi executada em uma janela de 600 por 420 [23]. A utilização de uma câmera com entrada USB comum, apesar de gerar certo atraso em relação a uma câmera que se utiliza diretamente do barramento, permite uma fácil substituição do modelo de câmera utilizado, facilitando-se assim upgrades de Hardware em futuras versões. Além disto as bibliotecas utilizadas para o *stream* da câmera possibilita também a utilização de câmeras via barramento – tal como a PiCamera – caso a velocidade transmissão se torne algo crucial em algum momento.

Figura 7 – WebCam Goldship 3817



Fonte:<<https://www.francavirtual.com.br/informatica/web-cam-s/web-cam-com-hub-usb-multimidia-goldship-3817>>

As características do modelo selecionado estão listadas na Tabela 5 a seguir.

Tabela 5 – Características técnicas WebCam Goldship 3817

Variável	Range
Sensor	CMOS 1.3M pixels
Resolução Máxima	4M pixels (interpolada)
TX Quadros	30 fps
Foco	5cm - Inf

Fonte:<<https://www.francavirtual.com.br/informatica/web-cam-s/web-cam-com-hub-usb-multimidia-goldship-3817>>

3.1.4 Mini Microfone USB

Para a integração do serviço de comando de voz foi necessário utilizar o Super Mini USB 2.0 Microphone, Figura 8. O dispositivo é considerado o menor do gênero no mundo, além disso possui a característica de ser facilmente instalado devido a sua conexão plug and play e seus drivers gratuitos. Em relação aos ruídos do ambiente, o microfone possui filtros que cancelam essas oscilações, mas na prática observa-se um leve ruído nas gravações de áudio realizadas [24]. Apesar disso, a API do Google Assistant consegue absorver as informações do áudio de forma eficiente.

Figura 8 – Super Mini USB 2.0 Microphone



Fonte:<https://mixvix.com.br/imprimir_a_nuncio-preco_722477>

As especificações estão listadas na Tabela 6 a baixo.

Tabela 6 – Características técnicas Super Mini USB 2.0 Microphone

Variável	Range
Sensibilidade	-67dBV / pBar, -47dBV / Pascal +/- 4dB
Resposta de Frequência	100 - 16kHz
Tensão de Operação	4.5V
Peso	3g

Fonte:<https://mixvix.com.br/imprimir_a_nuncio-preco_722477>

3.1.5 Servo Motor

Para o controle posicional do MUSK foi necessário a utilização do Servo TowerPro MG995, Figura 9, uma vez que ele possui uma alta qualidade, precisão, dimensão e facilidade no controle. As características principais estão sendo amostrados na Tabela 7 seguir.

Tabela 7 – Características técnicas TowerPro MG995

Parâmetro	Valor
Tensão de Operação	4,8 a 7,2V
Temperatura de Operação	-20°C a +60°C
Torque	13,0 Kg.cm (4,8V) e 15,0 Kg.cm (6,0V)
Velocidade	0,17 seg/60° (4,8V) e 0,14 seg/60° (6,0V)
Peso	69g
Dimensões	40 x 19 x 43 mm
Faixa de Rotação	180°
Tamanho do cabo	30 cm

Fonte:<https://mixvix.com.br/imprimir_a_nuncio-preco_722477>

Figura 9 – Servo motor TowerPro MG995

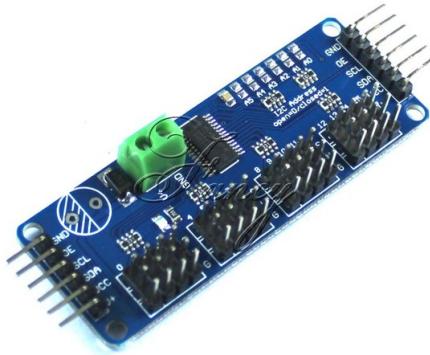


Fonte:<<https://www.filipeflop.com/produto/servo-towerpro-mg995-metalico/>>

O servo motor é um dispositivo eletromecânico controlado a partir da Modulação da Largura de Pulso (ou *Pulse Width Modulation* - PWM), onde o ângulo do motor é proporcional ao *Duty Cycle* do sinal. Para gerar o pulso PWM e utiliza-se o módulo PCA9685 da Adafruit, Figura 10 de baixo custo que estabelece a comunicação dos servos com a RPi através do protocolo [25].

A vantagem principal desse tipo de comunicação é o fato dele necessitar apenas dos pinos SDA e SCL para monitorar a ação dos servos, sendo que o número de periféricos que podem ser conectados ao módulo é de 16, logo se o projeto exigisse mais motores estes poderiam ser conectados e controlados facilmente com o PCA9685, sem prejudicar a estabilidade ou velocidade de processamento do sistema.

Figura 10 – PCA9685 Adafruit



Fonte:<<https://www.adrobotica.com/produto/servo-driver-16-canais-pca9685/>>

3.2 Sistemas Operacionais

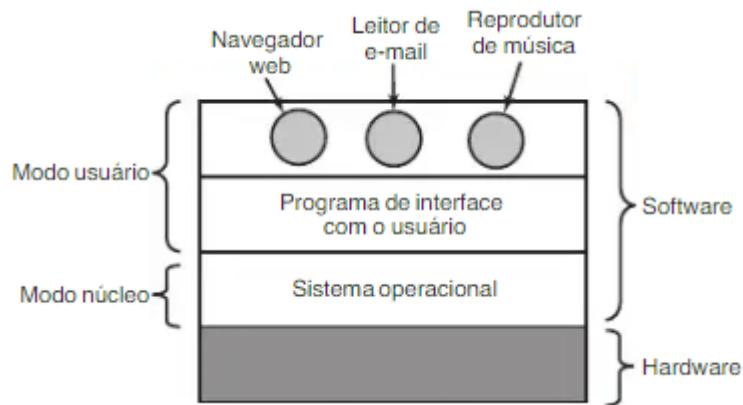
A definição de sistema operacional (SO) é algo um pouco impreciso [26]. A definição mais difundida é de que o SO consiste em um programa, ou conjunto de programas, responsáveis pelo gerenciamento dos recursos do computador, além de implementar uma interface entre máquina e o usuário através da comunicação com a camada de aplicação. Mais especificamente, o SO tem como função [26]:

- **Gerenciamento de memória:** permite que o usuário acesse de forma segura a memória durante suas requisições, além de garantir que cada aplicação tenha seu endereço próprio;
- **Gerenciamento de processos:** possibilita a impressão de que as tarefas estão sendo executados simultaneamente, no entanto o que ocorre é um partição do tempo para cada processo a partir do scheduler (escalonador) causando essa falsa concorrência;
- **Gerenciamento de recursos:** responsável por controlar o acesso das aplicações aos recursos do sistema, de forma eficiente, evitando a ocorrência de deadlocks;
- **Controle de fluxo de dados:** abstrai a complexidade da interface do hardware dos dispositivos periféricos, além de apresentar uma interface amigável para o usuário controlar de forma segura os dados;
- **Controle do sistema de arquivos:** controla o armazenamento e recuperação de informações de modo constante.

Outra característica importante do SO a ser ressaltada é a sua operação em modo núcleo, onde possui acesso completo a todo o hardware e pode executar qualquer instrução existente na capacidade da máquina. Em contrapartida os demais softwares atuam em modo usuário, onde possuem um subconjunto limitado de instruções da máquina disponível [26].

A Figura 11 ilustra de forma clara a disposição em camadas do sistema operacional dentro do computador.

Figura 11 – Disposição do sistema operacional no computador



Fonte: Autoria Própria

3.2.1 Raspbian

O Raspbian é uma variante não oficial do Debian (Wheezy armhf), sendo esta otimizada para códigos “hard float”. Desta forma possibilitando uma performance muito mais veloz para aplicações que fazem o uso intensivo de operações aritméticas com ponto flutuante e com as instruções ARMv6 [16].

Esta distribuição é fornecida gratuitamente, sendo inicialmente desenvolvida por Mike Thompson e Peter Green juntamente com entusiastas da placa Raspberry Pi [27]. O Raspbian tem como o seu ambiente padrão de desktop o LXDE, como gerenciador de janelas o OpenBox e navegador Chromium. Disponibilizado pela fundação Raspberry Pi este conta com uma versão do Mathematica e de uma versão para a Raspberry do jogo Minecraft.

Em função das necessidades deste projeto a versão utilizada do Raspbian foi uma imagem fornecida pela *AIY Projects – Google*. A qual por sua vez tem como base a versão *Stretch*, contando porém com algumas alterações. Entre elas a instalação das bibliotecas do Google Assistant e algumas modificações nos arquivos responsáveis pelas configurações da Raspberry. [27] [28]

Estes arquivos foram novamente modificados durante o projeto para que fosse

possível sobrescrever os requisitos de hardware fornecidos pela *AIY Projects – Google*. Permitindo assim que o software de reconhecimento de voz fosse utilizado dentro dos nossos parâmetros de hardware e software.

3.3 Estrutura de Software

A partir dos componentes de hardware selecionados foi integrado elementos de software no desenvolvimento do sistema embarcado. O assistente pessoal faz uso de linguagens, como Python, SQL, HTML, CSS e Javascript, frameworks, bibliotecas de visão computacional e web services. Assim, essa sessão tem como finalidade apresentar as estruturas de software utilizadas pelo MUSK.

3.3.1 Python

Para realizar a integração de todos os elementos de software presentes no projeto é utilizada o Python. Essa linguagem é uma das mais populares no mercado, devido ao seu alto nível, a sua versatilidade, sua curva de aprendizagem e a facilidade da leitura do código. Suas aplicações abrangem diversas áreas da computação: Backend, criação de CGIs para páginas dinâmicas scripts de execução e testes, processamento e análise de dados e principalmente machine learning.

A sua versatilidade é resultado da sua característica multiparadigma. O Python permite a programação utilizando conceitos de orientado a objetos, funcionais, imperativos e procedural. Além disso é uma linguagem de tipagem dinâmica, ou seja, que se adapta de acordo com o intuito e paradigma da aplicação. Ainda, por ser uma linguagem interpretada faz uso de programas chamados de interpretadores que realizam a leitura do código e a tradução para uma estrutura familiar ao sistema que irá executá-la. [29]

3.3.2 Web Server

O servidor Web tem como propósito o gerenciamento das requisições e respostas que ocorrem durante a interação do cliente com a aplicação no servidor. Assim, este é responsável pelas abstrações que tornam possível o envio de informações hipertexto.

Para realizar a comunicação entre os dois lados é utilizado o protocolo HTTP, que possibilita a estruturação das informações que serão transmitidas. O conteúdo de uma requisição ao servidor possui o tipo do método HTTP utilizado, o identificador da página que será acessada e os parâmetros do processo. Já o conteúdo da resposta possui o status da requisição, o tipo e o corpo do conteúdo a ser enviado para o cliente [30].

Os principais métodos do protocolo HTTP estão descritos a seguir [31].

- **GET**: solicita a transmissão de uma informação específica do servidor;
- **POST**: submete um corpo de dados já codificados para processar no servidor;
- **PUT**: envia arquivos para atualizar o servidor;
- **DELETE**: possibilita a exclusão de arquivos no lado do servidor.

3.3.2.1 Flask

Para o desenvolvimento do servidor Web responsável por controlar e fornecer algumas funcionalidades do assistente foi necessário determinar um ambiente de trabalho leve e simples para embarcar na Raspberry. Assim foi definido a utilização do Flask.

Segundo sua documentação, Flask é um micro-framework para aplicações Web que possui um núcleo simples e extensível, baseado nas bibliotecas WSGI *Werkzeug*, que é a interface padrão entre a aplicação Web python e o servidor HTTP para o desenvolvimento e aplicação, e *Jinja 2*, que é responsável pela renderização dos templates [32].

Uma das grandes vantagens do Flask é flexibilidade e capacidade de adaptação, suportando extensões para adicionar funcionalidades específicas na sua aplicação como se ela tivesse sido implementada no próprio framework.

O Flask não possui uma camada de abstração de banco de dados, o que permite que o usuário decida a biblioteca que irá ser utilizada no sistema. Permite a implementação de ferramentas de manipulação de bancos complexas, tanto para relacional quanto para não-relacional [33].

3.3.3 Páginas WEB

As páginas web, popularmente conhecidas como “sites”, foram desenvolvidas com base nas três principais linguagens de WEB, que são HTML, CSS e JavaScript (jQuery e Ajax) [34][35]. Os sites foram posteriormente estilizados com a ajuda da ferramenta DreamWeaver fornecida em forma de teste por 15 dias pela Adobe Co.

A seguir se observará quais as funções de cada linguagem e ferramentas utilizadas ao longo deste processo. Serão também apresentados alguns exemplos úteis e fundamentais tanto para o entendimento da linguagem como da confecção do trabalho.

É importante se enfatizar que para cada página web é necessário a geração de um novo código, o que torna este trabalho consideravelmente grande.

3.3.3.1 HTML

A linguagem HTML – ou, *Hyper Text Markup Language* (Linguagem de Marcação de Hipertexto em português) - foi criada (ainda não era uma linguagem, mas um conjunto

de ferramentas) por Tim Berners-Lee em 1991 com o intuito de facilitar a disseminação de artigos científicos [36][37]. Porém devido a sua sintaxe amigável e a popularização da internet esta linguagem passou a ser a principal linguagem web e vem constantemente se atualizando.

Atualmente o HTML encontra-se em sua versão 5.2, sendo que ao passar do tempo a sua linguagem passou a se tornar mais rígida. Atualmente o HTML tem como a principal função apresentar – ou exibir – e organizar a informação de determinado site. Para tal a semântica do HTML é dividida em *tags* – ou marcações – as quais são iniciadas e terminadas em parênteses angulares (e.g.: < >).

Na Figura 12, pode se observar o cabeçalho retirado da página “home_inside.html” na qual é possível atentar-se para a formatação básica de um código a qual conta com uma *tag*, *head*, *html*, *body* e por fim algumas *tags* de descrição, documentação e compilação – e.g.: “!doctype”.

Figura 12 – Estrutura HTML

```
<!doctype html>
<html lang="pt-br">
<head> . . . </head>
<body> . . . </body>
</html>
```

Fonte: Autoria Própria

A *tag html* delimita o inicio e o fim dos códigos em HTML. A *tag head* é responsável por conectar, organizar e apresentar certos arquivos, funções e informações que não são visíveis diretamente na página [36]. Alguns dos exemplos de informações contidas no *head* é o título da página, links para arquivos CSS e JavaScript além de informações da compilação do texto presente no *body* – UTF-8 –.

Já dentro da *tag body* está presente toda a informação que ficará visível na tela principal do usuário. Tal como títulos, subtítulos, texto, legenda, imagens e etc [36].

3.3.3.2 CSS

CSS ou, *Cascade Style Sheets*(Folha de Estilos Em Cascata) é uma linguagem utilizada juntamente com o HTML e tem como o principal intuito melhorar a apresentação da página, trabalhando sobre os seus designs, suas fontes, cores de fundo, entre outros[38].

Existem algumas maneiras diferentes de se utilizar o CSS ao longo de uma página HTML. A maneira profissional, é gerar uma biblioteca, sendo que no *head* do código em HTML existe uma referência para tal arquivo e ao longo do *body* é possível se referenciar a tais estilos a partir do atributo *class* [38]. Um exemplo pode ser observado na Figura

13 e 14.

Figura 13 – Estrutura HTML (1)

```
<head>
  <link href="css/simpleGridTemplate.css" rel="stylesheet" type="text/css">
</head>
```

Fonte: Autoria Própria

Figura 14 – Estrutura HTML (2)

```
<body>
  <div class="thumbnail">
    <a href="x" ></a>
  </div> \newline
</body>
```

Dentro do arquivo .css introduzido no *head* tem-se o seguinte modelo, ilustrado na Figura 15.

Figura 15 – Estrutura HTML (3)

```
.thumbnail { \newline
  width: 25%; \newline
  text-align: center; \newline
  float: left; \newline
  margin-top: 35px;
}
```

Fonte: Autoria Própria

É possível ainda alterar-se determinadas características de uma classe com o atributo *style* o que permite a se utilizar uma mesma classe de diferentes maneiras.

3.3.3.3 JavaScript

Dentre as linguagens WEB o JavaScript é a linguagem mais próxima a aquelas já vistas ao longo do curso, tal como C. O JavaScript é uma linguagem de programação interpretada. Em outras palavras o JavaScript tem como função automatizar determinados comportamentos da página quando acionado.

É importante também salientar que o JavaScript é uma linguagem de programação cliente-servidor. Este comportamento por sua vez abre espaço para scripts os quais funcionam de maneira muito semelhante à funções secundárias no C.

O JavaScript pode estar presente em uma biblioteca separada e ser chamado a partir do *head* HTML, conforme pode ser visto no código apresentado na Figura 16 seguir.

Figura 16 – Estrutura HTML com componente de javascript

```
<head>
|   <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="text/javascript"></script>
</head>
```

Fonte: Autoria Própria

As funções em JavaScript também podem ser escritas diretamente dentro da *tag* script dentro do header, esta utilização se apresentou como a mais interessante neste trabalho pois foram necessárias poucas requisições entre cliente e servidor.

3.3.4 Banco de Dados

Segundo Korth, um banco de dados (BD) pode ser definido como:

"Uma coleção de dados inter-relacionados, representando informações sobre um domínio específico." [39]

Logo, ao se agrupar informações que se relacionam e tratam de um assunto em comum, têm-se um BD. Esta ferramenta é essencial no ramo empresarial, uma vez que facilita o serviço de sistema de informações.

O sistema de gerenciamento de banco de dados (SGBD) é uma ferramenta capaz de manipular informações do banco de dados e de interagir com o usuário. Um sistema de BD pode ser subdividido em: dados, hardware, software e usuários. Segundo Date:

"Um sistema de bancos de dados pode ser considerado como uma sala de arquivos eletrônica." [40]

A função primordial de um SGBD é de facilitar a interação entre o usuário e os dados, uma vez que este não necessita analisar os detalhes internos do BD. Além disso, a ferramenta torna independente os dados para com as aplicações, permitindo assim um alto nível de abstração, principalmente por parte do utilizador do serviço.

Os bancos de dados são portanto extremamente difundidos e presentes nas mais diferentes aplicações. Com o intuito então de se adaptar a cada aplicação dois modelos de bancos de dados foram desenvolvidos, são estes:

- **Banco de Dados Relacional:** Este modelo de banco de dados trata e apresenta as informações em forma de Tabela, ou em outras palavras a partir de relações. Esse

tipo de modelo começou a ser concebido na década de 1970 e tinha como principal objetivo simplificar a estruturação de soluções permitindo que um grande conjunto de dados pudesse ser analisado e co-relacionado de maneira simples e organizada[41].

- **Banco de Dados Não Relacional:** Diferente do modelo relacional este método não utiliza-se do esquema de organização em linhas e colunas. Para tratar de informações guardadas este tipo de modelo busca a otimizar a sua estrutura de acordo com a informação armazenada. Podendo assim ter seus dados armazenados tanto a partir de documentos JSON até gráficos definidos por vértices e bordas[42].

3.3.4.1 SQLite

No escopo do projeto utiliza-se a biblioteca SQLite no armazenamento de tarefas do usuário para eventuais consultas. Essa biblioteca está presente em grande parte das distribuições Linux, escrita em C, que permite a criação, edição e operação dentro de bancos de dados SQL. É importante ressaltar que diferente do MySQL e outras bibliotecas o principal objetivo do SQLite não é se conectar a servidores externos de bases de dados, mas sim ao servidor local.

A linguagem SQL é dividida em conjuntos que são relacionados ao tipo de operação que deseja se realizar no banco. Esses grupos são apresentados e exemplificados a seguir.

- **Linguagem de Manipulação de Dados (DML):** permite a inclusão (*INSERT*), atualização (*UPDATE*) e exclusão (*DELETE*) de dados presentes em diversas Tabelas do banco ao mesmo tempo;
- **Linguagem de Definição de Dados (DDL):** permite a definição de novas Tabelas e elementos associados. Os comandos presentes nesse grupo são de criação (*CREATE*), exclusão (*DROP*) e alteração (*ALTER*) do banco de dados ;
- **Linguagem de Controle de Dados (DCL):** responsável pelo controle de permissões dos dados e usuários que tem acesso para consultar ou manipular as Tabelas do banco de dados. Para habilitar o acesso é utilizado o comando *GRANT* e para desabilitar o *REVOKE*;
- **Linguagem de Transação de Dados (DTL):** caracteriza o controle das transações executadas no banco. A instrução *BEGIN TRANSACTION* inicia o processo, o *COMMIT* indica a finalização da transação e o *ROLLBACK* descarta as mudanças da última alteração;
- **Linguagem de Consulta de Dados (DQL):** responsável pela operação de consulta de dados presentes no banco. A instrução mais utilizada é o *SELECT* .

O SQLite é, principalmente, indicado para aplicações com baixas quantidade de acessos e requisições, além é claro de sistemas com baixa concorrência. Delimitando-se, portanto, a aplicações IoT - como é o caso deste projeto -, aplicações Desktop e aprendizado de banco de dados. [43]

3.3.5 Interface Homem-Máquina

Essa sessão apresenta os recursos utilizados para proporcionar uma interface homem-máquina (HMI) prática, flexível para futuros desenvolvimentos e confiável nos resultados gerados.

3.3.5.1 Interface Gráfica

Um dos elementos mais importantes na transmissão de informações entre usuário e máquina se dá através da interpretação visual. Desta forma a presença de uma tela mostrou-se como uma opção interessante para que houvesse uma relação mais eficiente na interface homem-máquina (ou *Human-Machine Interface* - HMI). Portanto utilizou a biblioteca de desenvolvimento visual chamada kivy.

O Kivy é uma framework de código aberto, multiplataforma, criado para a geração de aplicações com componentes visuais em interfaces hodiernas, contando também com suporte a “touch e multi-touch”. O principal diferencial do Kivy é o seu apelo para um rápido desenvolvimento e fácil interação com design.

Escrito em Python e Cython o Kivy é baseado em OpenGL ES 2 e é suportado em uma grande variedade de sistemas, além de possuir uma documentação e biblioteca extensa e variada. A grande vantagem do Kivy é que assim como o Python ele pode ser compilado em diversas plataformas, gerando assim uma maior flexibilidade para os códigos.

3.3.5.2 Visão Computacional

Segundo o Data Science Academy, a visão computacional é o processo de modelagem e replicação da visão humana usando software e hardware. Assim o estudo dessa área está vinculado aos avanços de algoritmos e métodos de machine learning, uma vez que busca transmitir as informações do ambiente de forma comprehensível para o computador [44].

Entre as tarefas típicas de visão computacional, pode-se citar [45]:

- Reconhecimento;
- Identificação;

- Detecção;
- Movimento;
- Reconstrução de cena;
- Restauração de imagens.

Uma das funcionalidades integradas ao projeto é a capacidade do assistente pessoal reconhecer pessoas no ambiente em que esta monitorando. Para realizar tal feito é necessária a utilização das tarefas de detecção e reconhecimento em imagens.

O processo de reconhecimento facial possui, de forma geral, três etapas. Inicialmente realiza-se a detecção de faces dentro de imagens ou vídeos a partir da busca por formas geométricas semelhantes ao rosto humano, além de eliminar do processamento informações irrelevantes presentes na entrada. Em seguida é realizado a extração dos pontos nodais, ou seja, as características presentes no rosto como por exemplo a distância entre os olhos, o comprimento do nariz, o tamanho do queixo e a linha da mandíbula. Esses dados podem ser armazenados em um banco de dados para realizar um refinamento no algoritmo de reconhecimento ou apenas de forma temporária, uma vez que o algoritmo já está consolidado. Após a definição do conjunto de dados característicos é efetuada a identificação da persona, a partir dos dados persistentes obtidos no treinamento do modelo [46].

3.3.5.2.1 OpenCV

Acrônimo para Open Source Computer Vision, o OpenCV é uma biblioteca de programação de código aberto. Nascido de um projeto da empresa Intel Research o OpenCV tinha como ideia inicial fornecer serviços para o aprimoramento de aplicações de elevado uso de CPU[47][48]. Ao longo dos anos o OpenCV contou com várias versões, sendo que a cada nova versão ficava mais notável a sua afinidade com o processamento de imagens.

O OpenCV é requisitado no escopo do projeto para compor a funcionalidade de reconhecimento facial. No entanto a biblioteca é responsável por apenas realizar detecções faciais dentro do ambiente que está sendo monitorado pelo assistente pessoal. A versão utilizada neste projeto é o *"OpenCV 4.0.0 Light"*, uma versão modificada que apesar de ter menos funções é a versão mais adequada às características de processamento da Raspberry Pi 3B+. Na versão atual as suas funções mais evidentes são [48]:

- **Conjunto de Ferramentas para imagens 2D e 3D:** Análise de imagens (histograma, entre outros), alteração de sistema de cores, alteração da imagem (e.g.: desenhar formas sobre a imagem), etc;

- **Movimentação Espacial da Câmera (EgoMotion):** Capacidade de percepção de movimento da câmera;
- **Sistema Reconhecimento Facial:** Capacidade de comparar formas e reconhecer rostos;
- **Interação Homem-Máquina:** Capacidade de gerar janelas e outros para a interação com o usuário;
- **Segmentação e Reconhecimento:** Capacidade de definir e reconhecer diferentes objetos. É importante porém ressaltar que a utilização desse recurso demanda muito da Raspberry e por isso não demonstrou ser uma característica viável.

3.3.5.3 Amazon Web Services

Amazon Web Services (AWS) é uma plataforma de serviços de computação em nuvem que oferece entrega sob demanda de poder computacional, armazenamento de banco de dados, aplicações e outros recursos de TI pela internet. Dentre suas principais características pode-se citar [49]:

- **Elasticidade e Escalabilidade:** é a capacidade do ambiente computacional da nuvem aumentar ou diminuir de forma automática os recursos computacionais demandados e provisionados para cada usuário;
- **Confiabilidade:** alto desempenho no processamento e transmissão de dados, tolerância a falhas;
- **Zonas de Disponibilidade:** são Centros de Processamento de Dados (ou *Data-centers*) separados e isolados fisicamente, que são conectados com baixa latência, alta taxa de transferência e redes altamente escaláveis;
- **Pay-as-you-go:** paga apenas pelos serviços individuais que precisar, pelo tempo que os utilizar, sem a necessidade de contratos de longo prazo ou licenciamento complexo;

3.3.5.3.1 AWS Rekognition

Para o desenvolvimento da funcionalidade de reconhecimento facial do assistente empregou-se o *Rekognition*. Este é um dos diversos serviços disponibilizados e gerenciados pela AWS, que está dentro da área de visão computacional. Sua aplicação facilita a análise de imagens e vídeos a partir de APIs próprias, de forma escalável e eficiente. Os tipos de inferências que podem ser realizados está listado a seguir [50].

- **Rótulos:** objetos, eventos, conceitos (noite, natureza) e atividades;
- **Faces:** detecta faces em imagens/vídeo, extraíndo informações como pontos de referência faciais, emoções, onde as faces são detectadas e comparação entre imagens;
- **Pesquisa de face:** as informações faciais são indexadas em um contêiner que são utilizados como uma coleção de base;
- **Caminhos das pessoas:** Monitora o caminho de pessoas detectadas em um vídeo armazenado. Fornece informações de rastreamento de caminho. Detalhes da face e localização de quadros das pessoas detectadas;
- **Detecção de texto:** Detecta texto em imagens e os converte em textos legíveis por máquinas;
- **Outros:** reconhecimento de celebridades/conteúdo desprotegido.

O serviço ainda permite configurar o seu modo de operação: com e sem armazenamento de dados. No primeiro, o Rekognition recebe os dados de entrada (imagem ou vídeo), analisa e retorna os resultados, mas nada é salvo na nuvem. O segundo, realiza a análise dos dados, devolve os resultados e os deixa armazenado para futuras ações.

3.3.5.4 Comando de Voz

Uma das principais habilidades do MUSK é a capacidade de interagir com o usuário a partir de um sistema de comandos de voz, no entanto existem algumas dificuldades para se trabalhar de forma adequada com sinais de áudio. Para se obter resultados eficientes é necessária a utilização de alguns métodos, via software e hardware, durante todo o processo.

Inicialmente é realizada a digitalização do sinal de áudio a partir de um conversor analógico-digital que gera dados digitais através das vibrações produzidas. Em seguida é realizado uma filtragem para a retirada de ruídos e interferências. Então é efetuado um processo para a obtenção da característica espectral no domínio da frequência do sinal digital, aplicando a Transformada Rápida de Fourier. Após a aplicação do método de Fourier ocorre a separação do sinal digitalizado em partes ainda menores (sons fonéticos), não maiores que uma sílaba. O software compara os sons captados com os fonemas já conhecidos e persistidos no banco de dados que correspondem com o idioma do locutor. A última etapa utiliza palavras e frases já conhecidas para comparar com os resultados obtidos no processo anterior e converte o comando para a funcionalidade vinculada [51].

Essa tecnologia possibilita a implementação e integração de diversas aplicações que a utilizam como gatilho para a execução de ações. Como exemplo de aplicação, pode-se

citar o seu uso para automação residencial, escritórios inteligentes, atendimento eletrônico (operadoras, bancos e serviços do governo) e assistentes pessoais.

Assim, para a funcionalidade de comando de voz é utilizado um serviço existente e consolidado no mercado: Google Assistant. Pois o intuito do projeto não é o desenvolvimento de um sistema de comando de voz (que envolve todo o procedimento apresentado anteriormente), mas sim de um assistente pessoal que integra diversas funcionalidades.

3.3.5.4.1 Google Assistant

O Google Assistant é um assistente pessoal virtual desenvolvido pelo Google e que se assemelha com outros concorrentes no mercado como Alexa (Amazon), Siri (Apple) e Cortana (Microsoft). Este possui a habilidade de realizar diversas tarefas do dia-a-dia, como ligar para uma pessoa, mandar mensagem, realizar pesquisas, além de interagir com o usuário.

O sistema ainda possui compatibilidade com diversas plataformas e hardwares, como:

- Google Home;
- Celulares Android;
- Tablets;
- Iphone e Ipad;
- Raspberry Pi.

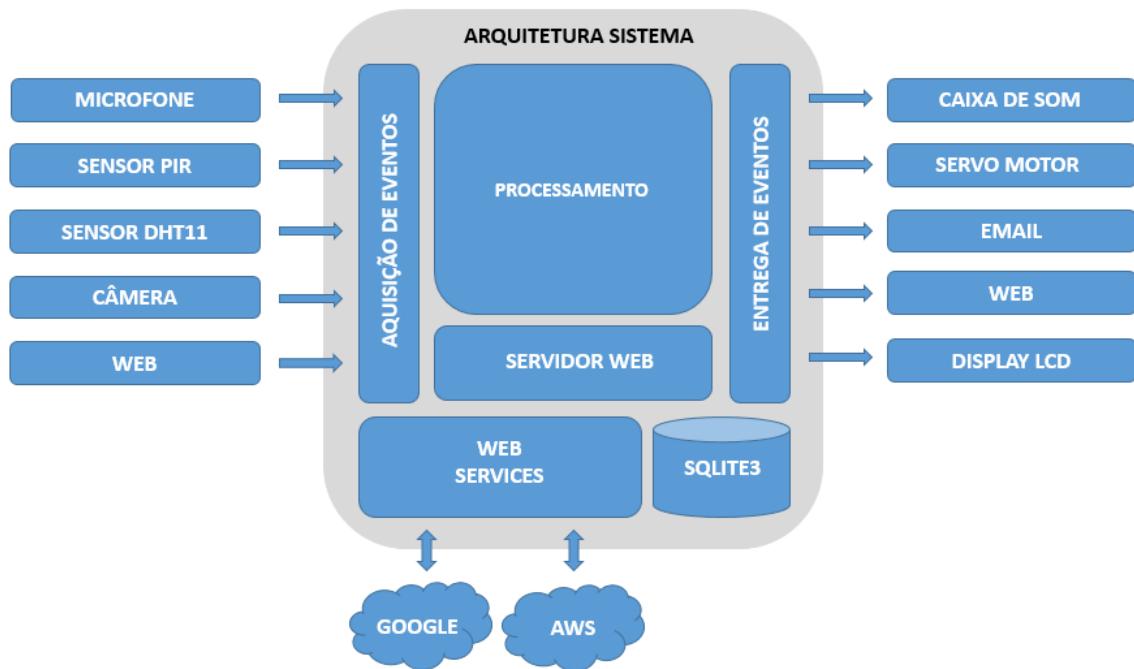
Além das funcionalidades já integradas, existe a possibilidade de desenvolver novas habilidades e integrá-las ao sistema. Essas funções são chamadas de *actions* que são o ponto de entrada em uma interação entre o usuário e o Google Assistant, criadas através de linguagem natural, em que o usuário pode falar ou digitar uma frase informando o nome da *action* que deseja interagir [52].

4 Desenvolvimento

4.1 Arquitetura do Sistema

A Figura 17 a baixo ilustra o esquemático da arquitetura do sistema.

Figura 17 – Arquitetura do Sistema



Fonte: Autoria Própria

Os elementos presentes na arquitetura final do MUSK, são descritos a seguir:

- **Processamento:** Desenvolvido através da linguagem Python e suas bibliotecas. Responsável por gerenciar as ferramentas, executar o servidor Web e serviços do sistema;
- **Servidor Web:** Desenvolvido através do framework Flask. Responsável por executar o servidor e de fornecer ferramentas para gerenciar as requisições;
- **Web Services:** Referente as bibliotecas utilizadas para integrar o sistema com APIs dos serviços Web, AWS e Google Cloud;
- **SQLite3:** Responsável por fazer a comunicação do sistema com o banco de dados;
- **Aquisição de Eventos:** Responsável por estabelecer a comunicação entre elementos externos e o sistema;

- **Entrega de Eventos:** Responsável pelos dados de saída do sistema para os dispositivos e serviços periféricos.

Nos próximos tópicos serão detalhados os elementos que compõem o assistente pessoal e os processos utilizados no desenvolvimento dos mesmos, de forma a possibilitar a reprodução deste trabalho.

4.2 Configuração da Raspberry PI

Para o desenvolvimento do projeto foi selecionado a RPi apresentada na sessão anterior. Uma vez que essa não possui um SO embutido e nem uma memória de massa, foi necessário utilizar um cartão micro SD, ilustrado na Figura 18.

Figura 18 – Cartão micro SD



Fonte:<<https://www.magazineluiza.com.br/cartao-de-memoria/informatica/s/in/ctdm/>>

Então, foi realizado o download da imagem do SO Raspbian e instalado no cartão, através do software Etcher.

O Raspbian assim como grande parte dos sistemas operacionais tem por padrão uma aquisição dinâmica de IP, porém para que seja possível uma comunicação entre os computadores dentro desta rede com a placa em questão é necessária que se conheça o seu IP. Para tal é necessário que haja uma alocação estática de IP.

A alocação estática de IP será dada a partir do roteador o qual determinará um IP com base no endereço de MAC da RPi. Para tal o roteador comparará os endereços MACs presentes na rede LAN e caso haja uma convergência o mesmo estabelecerá uma conexão a partir do endereço de IP presente no IPtable.

Duas técnicas foram utilizadas para a alteração do endereço MAC da RPi. O primeiro contou com a utilização do serviço de DHCP e a alteração do arquivo responsável

por configurar as conexões de internet. A seguir está presente o passo a passo desta técnica. [53] [54]

Edição do Arquivo Interfaces: Para editar o arquivo interfaces é necessário a permissão de administrador, assim o comando a seguir foi utilizado.

```
sudo gedit /etc/network/interfaces
```

Determinou-se a conexão automática e prioritária via eth0 – Conexão Física – e o uso do DHCP.

```
Auto eth0
iface eth0 inet dhcp
```

Adicionou-se o MAC Address desejado, para que fosse possível haver um endereço fixo caso o modem esteja preparado para isso.

```
hwaddress ether 01:02:03:04:05:06
```

Reiniciou-se a rede e a RPi para que tais medidas fizessem efeito.

```
sudo /etc/init.d/networking restart
sudo shutdown -r now
```

O segundo método utilizado foi o spoofing – “falsificação” - do endereço MAC. Para tal seguiu-se o método a seguir.

Editou-se o arquivo `systemd-network`, conforme presente no comando a seguir.

```
sudo gedit /etc/systemd/network/00-defalt.link
```

Determinou-se o MAC address desejado, Figura 19, a partir do spoofedMAC: [55]

Figura 19 – Configurações no MAC

```
[Match]
MACAddress=original MAC

[Link]
MACAddress=spoofed MAC
NamePolicy=kernel database onboard slot path
```

Fonte: Autoria Própria

Ainda para que seja possível a conexão com outros computadores dentro da mesma rede é necessário que se habilite a conexão via SSH na RPi, para tal deve-se seguir os passos a seguir.

Habilitar o uso da SSH da Raspberry a partir do systemctl: [56]

```
sudo systemctl enable ssh
```

Iniciar o uso da SSH pela Raspberry:

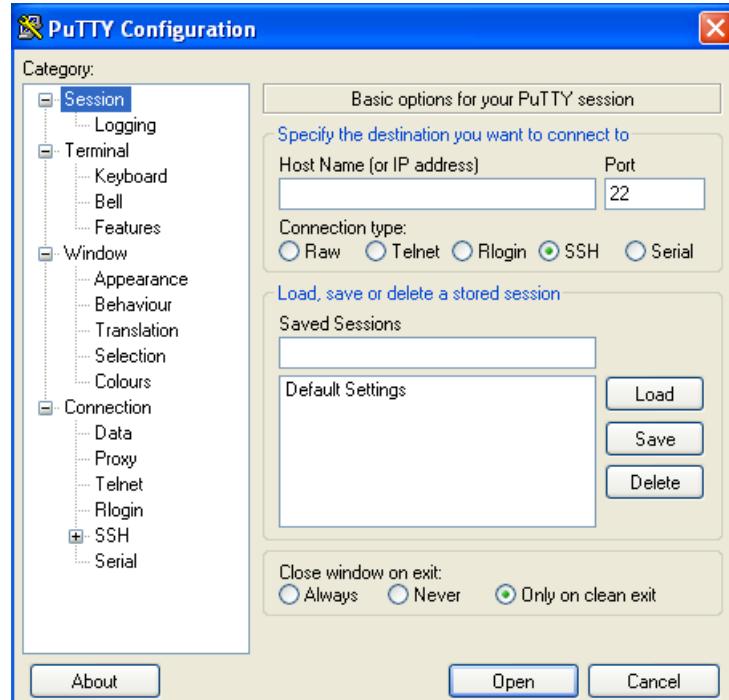
```
sudo systemctl start ssh
```

Para que um computador possa se conectar a raspberry agora basta caso este seja Linux dar o comando ssh no terminal. Como pode ser visto a seguir.

```
ssh root@70.32.86.175
```

Caso se esteja utilizando o Windows é necessário a utilização de algum programa auxiliar que suporte tal conexão. Neste trabalho foi-se utilizado o Putty, conforme visto na Figura 20.

Figura 20 – Terminal Putty



Fonte: Autoria Própria

Configurou-se então o Putty com o IP alocado pelo roteador presente no laboratório (IP: 10.235.10.48), determinou-se que a porta de conexão seria a 22.000 (vinte e dois mil) e a conexão via SSH.

4.3 Configuração do Servidor Flask

O servidor WEB é o servidor que será acessado pelo usuário para verificar e controlar as ações do MUSK. Para o projeto foi utilizado o WEB Server Flask, cuja função é possibilitar o acesso a página WEB construída para o assistente, a partir do gerenciamento de requisições e da manipulação dos dados. [57]

Inicialmente foi necessário verificar se o python instalado na RPi está atualizado a partir do comando:

```
sudo apt-get update
sudo apt-get upgrade
```

Em seguida, instalou-se o sistema de gerenciamento de pacotes (PIP) e então o Flask foi instalado, a partir dos seguintes comandos:

```
sudo apt-get install python-pip python-flask
sudo pip install flask
```

Foi então criado um diretório onde ficaram armazenados os arquivos de programa referentes ao projeto, a partir do comando:

```
sudo mkdir web-server
```

Dentro do diretório foi criado o arquivo app.py o qual será executado infinitamente, sendo este o responsável pelo gerenciamento do projeto. O comando utilizado foi:

```
sudo nano app.py
```

Para as páginas HTML, foi criado uma pasta dentro do repositório chamada 'templates'.

```
sudo mkdir templates
```

Nesta foi armazenado todos os códigos em HTML elaborados no projeto. A parte de elaboração dos códigos WEB será apresentada a seguir.

4.4 Web

A interface WEB tem um importante papel no projeto do assistente pessoal, já que é a partir dela que os usuários são capazes de ter acesso remoto aos dispositivos presentes em suas casas e - ou - escritórios. Para o desenvolvimento da interface WEB

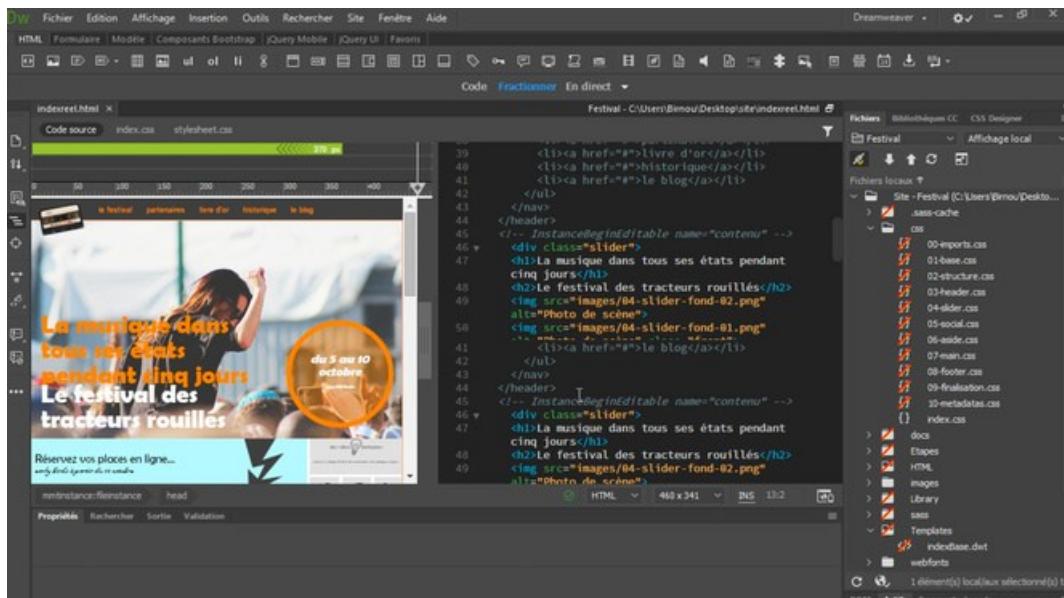
foram utilizadas as linguagens HTML, CSS e JavaScript. Com o intuito de aumentar a velocidade do desenvolvimento utilizou-se da ferramenta da DreamWeaver da Adobe. A qual fornece um licença de 15 dias gratuitos de teste. Ao longo deste tópico será abordado o desenvolvimento da interface WEB a partir do DreamWeaver.

4.4.1 DreamWeaver

O DreamWeaver é uma plataforma de desenvolvimento web baseado em WYSIWYG (*What You See Is What You Get* – O Que Você Vê É O Que Você Têm, em português) que possui suporte a várias tecnologias WEB, como HTML, XHTML, CSS, JavaScript, Ajax, PHP, ASP, ASP.NET, JSP, entre outros [58].

O DreamWeaver conta tanto com uma IDE que permite combinar modos de edição visual quanto de código agilizado – sendo este o método utilizado neste trabalho. O DreamWeaver possui funções de Autocomplete e visualização dinâmica as quais permitem a produção mais veloz de páginas web. Pode-se observar a sua interface na Figura 21.

Figura 21 – Interface DreamWeaver



Fonte: Autoria Própria

4.4.2 Página Inicial e de Aplicativos

O desenvolvimento da tela inicial e de aplicativos se deu para que a utilização do site ocorra de forma orgânica. Sendo que o seu principal objetivo é fornecer uma porta de entrada para as funções mais internas do projeto.

O seu desenvolvimento contou com fontes e estilo JavaScript de acesso livre fornecidas pela adobe, com o estilo de CSS conhecido como *hero*. O código para a criação

de tal página está descrito no apêndice J, o resultado por sua vez pode ser observado na Figura 63.

4.4.3 Página Modo Vigia

A concepção desta página tem como objetivo servir como fonte de segurança para a casa. Permitindo que no momento em que um usuário inesperado for captado, um e-mail é mandado para o responsável cadastrado. Informando-o e possibilitando não apenas que este observe o cômodo, mas também move a câmera.

Sobre essa interface funciona um dos elementos chaves para o controle dos servos motores. Como pode ser observado na Figura 22, existe uma função Ajax (Java Script) que é acionada a cada vez que os botões são clicados.

Figura 22 – Função Ajax para o Controle dos servos Motores

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
<script language="javascript" type="text/javascript">
    function Button_onclick(direction) {
        $.ajax({ url: '/' + direction })
    }
</script>
```

Fonte: Autoria Própria

Cada vez que um dos botões é acionado um comando é enviado através da URL para o código em Python, o qual por sua vez interpreta qual a direção em que os servos motores devem seguir, conforme visto no apêndice L. Onde pode se observar cada botão envia uma “string” para a função que altera a URL de acordo com o botão acionado.

4.4.4 Página Sensoriamento (Temperatura e Umidade)

Quanto a esta página pode-se observar que o código em HTML recebe os dados enviados através do servidor Flask. Os dados são recebidos a partir de um dicionário que transmite um conjunto de strings e pode ser recebido e decomposto.

Figura 23 – Código HTML, CSS - Temperatura e Umidade

```
<div class="forecast-content">
    <div class="degree">
        <div class="num">{{ temp }}<sup>o</sup>C</div>
        <div class="forecast-icon">
            
        </div>
    </div>
</div>
```

Fonte: Autoria Própria

Como pode ser observado no código do apêndice M e nas Figuras 23 e 24, os dados são apresentados no site através de variáveis (temp e umid) com nomes iguais aos presentes no dicionário transmitido contidas em chaves duplas. [59]

Figura 24 – Código Python - Temperatura e Umidade

```
@app.route('/temperatura')
def temp():
    umidade, temperatura = sensor.readDht11()
    templateData = {'umid': "10", 'temp': "10"}
    return render_template('temperaturahtml.html', **templateData)
```

Fonte: Autoria Própria

Já o apêndice A apresenta a passagem do dicionário (“templateData”) que contém as variáveis a serem apresentadas no site.

4.4.5 Página Agenda

A página da agenda foi construída para que fosse possível se inserir tarefas e houvesse uma forma com a qual o usuário fosse capaz de visualizar e ser lembrado de tais tarefas. Assim para se iniciar um código periodicamente foi-se utilizado o “crontab”- presente em sistemas Linux, mas especificamente baseados em debian - no qual pode-se iniciar um programa que a partir da quantidade de tarefas presentes no banco de dados este gera uma Tabela dinâmica de botões correspondente a cada tarefa.

Sobre o código que controla a Interface da agenda web é interessante salientar primeiramente o método pelo qual são enviados novas tarefas – apêndice N - ao arquivo em python para que o mesmo salve as atividades em um banco de dados.

Como pode ser visto no apêndice N através de um método POST é mandado um comando para que o python passe do seu método atual para o seu método “add”, isso pode ser observado pelo comando presente no atributo “action” do código HTML. [60]

Como pode ser observado no apêndice ?? o método “add” recebe também um atributo “post”, o que permite que ele faça requerimentos de itens dentro de “forms” HTML. No caso em questão é passado o título da tarefa e a data da mesma. Em seguida esses dados são colocados na Tabela do banco de dados e a mesma é salva. [61]

Já a referência que indica se uma tarefa já foi completada ocorre através da alteração da URL entre tarefa completa ou não, conforme pode ser visto no código na Figura 25 seguir. Nesse código ainda é possível se observar comandos em chaves duplas para a geração de Tabelas dinâmicas e o recebimento de variáveis do código em python (todo.text, todo, incomplete, etc.).

Figura 25 – Código HTML – complete, Incomplete

```

<div>
  <h2>Incomplete Items</h2>
  <ul>
    {% for todo in incomplete %}
      <li style="font-size: 30pt">{{ todo.text }} <a href="{{ url_for('complete', id=todo.id) }}">Mark As Complete</a></li>
    {% endfor %}
  </ul>
  <h2>Completed Items</h2>
  <ul>
    {% for todo in complete %}
      <li style="font-size: 30pt">{{ todo.text }}</li>
    {% endfor %}
  </ul>
</div>

```

Fonte: Autoria Própria

Já no código em python, na Figura 26, pode-se observar que ao se completar uma tarefa o método “complete” recebe o id da tarefa em questão e altera o estado da coluna “complete” informando que a tarefa foi completada. O código indicando isso está presente no código a seguir.

Figura 26 – Método de Tarefa Completada

```

@app.route('/complete/<id>')
def complete(id):

    todo = Todo.query.filter_by(id=int(id)).first()
    todo.complete = True
    db.session.commit()

    return redirect(url_for('index'))

```

Fonte: Autoria Própria

4.5 Interface Gráfica

O desenvolvimento da interface gráfica como já introduzido anteriormente se deu pela utilização do framework Kivy. Desta forma para que haja um pleno funcionamento do Kivy é necessário primeiramente a sua instalação e a de suas bibliotecas adjacentes conforme poderá ser observado através dos passos a seguir.

Deve-se primeiramente atualizar a lista de pacotes, para tal utiliza-se o comando abaixo:

```
sudo apt-get update
```

Instala-se algumas dependências do Kivy:

```

sudo apt-get install kivy-examples
sudo apt-get install python-setuptools python-pygame

```

```
python-opengl  python-enchant python-dev build-essential python-pip libgl1-mesa-dev libgles2-mesa-dev zlib1g-dev
```

Instala-se a versão mais recente do Cython:

```
sudo pip install --upgrade Cython==0.28.2
```

Instala-se o kivy para Python 2:

```
sudo apt-get install python-kivy
```

Instala-se o kivy para Python 3:

```
sudo apt-get install python3-kivy
```

Por fim, instala-se alguns exemplos de aplicações:

```
sudo apt-get install kivy-examples
```

O funcionamento do kivy pode se dar basicamente de duas formas, uma destas é adicionando um arquivo com a extensão “.kv” sobre o mesmo diretório do arquivo python que está rodando no momento. Sendo que no arquivo python deve-se procurar acionar o programa a partir de uma classe instanciada com um argumento App da biblioteca kivy.app. Tal dinâmica pode ser melhor entendida a partir da Figura 27 a seguir.

Figura 27 – Exemplo arquivo Python

```
import kivy
kivy.require("1.10.0")

from kivy.app import App
from kivy.uix.floatlayout import FloatLayout

class FloatingApp(App):
    def build(self):
        return FloatLayout()

flApp = FloatingApp()
flApp.run()
```

Fonte: Autoria Própria

Figura 28 – Exemplo arquivo kivy

```
# ----- FLOATING.KV -----
# size_hint defines the widget width (0-1) representing
# a percentage of 100% and height of the window
<CustomButton@Button>:
    font_size: 32
    color: 0, 0, 0, 1
    size: 150, 50
    background_normal: ''
    background_down_color: 0, 0, 0, 1
    background_color: .88, .88, .88, 1
    size_hint: .4, .3

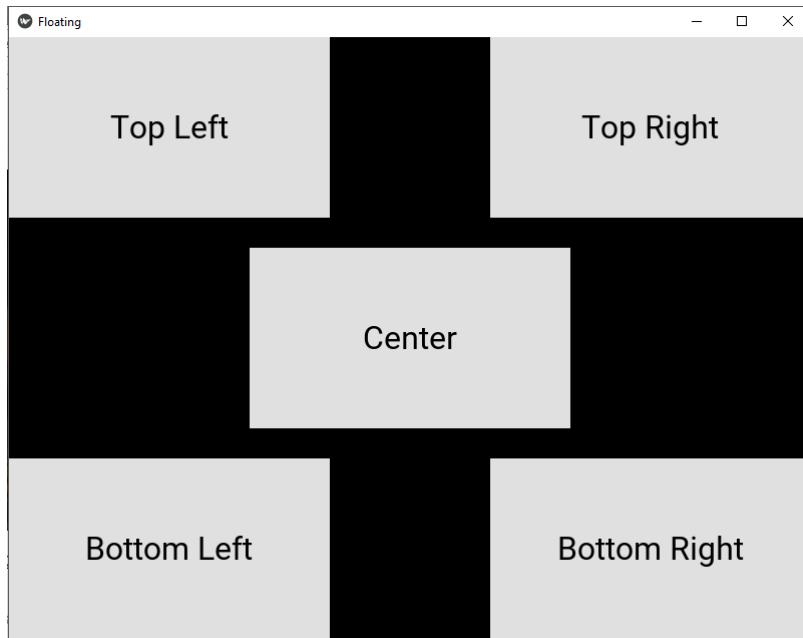
# pos_hint represents the position using either x, y, left,
# right, top, bottom, center_x, or center_y

<FloatLayout>:
    CustomButton:
        text: "Top Left"
        pos_hint: {"x": 0, "top": 1}
    CustomButton:
        text: "Bottom Right"
        pos_hint: {"right": 1, "y": 0}
    CustomButton:
        text: "Top Right"
        pos_hint: {"right": 1, "top": 1}
    CustomButton:
        text: "Bottom Left"
        pos_hint: {"left": 1, "bottom": 0}
    CustomButton:
        text: "Center"
        pos_hint: {"center_x": .5, "center_y": .5}
```

Fonte: Autoria Própria

Conforme pode ser visto na Figura 27 após ser acionado o arquivo Python busca instâncias apresentadas no arquivo kivy e que estejam presentes dentre as bibliotecas importadas para o arquivo Python. Ao se compilar o programa apresentado é gerada a tela presente na Figura 29.

Figura 29 – Tela kivy



Fonte: Autoria Própria

A segunda forma é a mais utilizada é a apresentada na Figura 30, onde é adicionada uma biblioteca builder. Permitindo assim que o arquivo kivy e o arquivo Python fiquem sobre um único arquivo Python. Tal código deverá gerar a mesma tela da gerada pelos códigos apresentados anteriormente.

Figura 30 – Exemplo código Python com kivy

```

# ----- FLOATING.KV -----
<CustButton@Button>:
    font_size: 32
    color: 0, 0, 0, 1
    size: 150, 50
    background_normal: ''
    background_down_color: 0, 0, 0, 1
    background_color: .88, .88, .88, 1
    size_hint: .4, .3

<FloatLayout>
    CustButton:
        text: "Top Left"
        pos_hint: {"x": 0, "top": 1}
    CustButton:
        text: "Bottom Right"
        pos_hint: {"right": 1, "y": 0}
    CustButton:
        text: "Top Right"
        pos_hint: {"right": 1, "top": 1}
    CustButton:
        text: "Bottom Left"
        pos_hint: {"left": 1, "bottom": 0}
    CustButton:
        text: "Center"
        pos_hint: {"center_x": .5, "center_y": .5} ```

class FloatingApp(App):
    def build(self):
        return FloatLayout()

f1App = FloatingApp()
f1App.run()

```

Fonte: Autoria Própria

4.5.0.1 Geração de Telas

Após a inicialização do assistente, a tela de entrada dele é vista conforme ilustrado na Figura 31. A interface fica alternando entre as imagens como uma forma de dinamizar o MUSK. Permitindo assim a sua função Touch (e Multi-Touch consequentemente) seja acionada em suas diversas telas.

Para produzir tal efeito foi utilizada a biblioteca kivy para o python. O código foi incorporado ao programa principal deste projeto, conforme o Apêndice A. As imagens que são vistas na tela são tratadas como botões que mudam em um determinado período de tempo.

A interface visual é dividida em três telas principais, que são elas:

- 1. Desbloqueio Inicial
- 2. Dados de Sensoriamento
- 3. Informações Pessoais

A tela de desbloqueio inicial pode ser observada na Figura 31, a qual tem como função servir como tela de descanso e futuramente exigir uma senha de entrada.

Figura 31 – Interface Kivy (1)



Fonte: Autoria Própria

Para a construção de tal tela utilizou-se os comandos apresentados em código kivy presentes na Figura 32.

Figura 32 – Código Kivy - Interface (1)

```
<ScreenOne@Button>:

    background_color: 0,0,0,1

    on_press:
        root.manager.current = 'screen_two'
    Image:
        source: 'skull1.jpg'
        y: self.parent.y + self.parent.height - 600
        x: self.parent.x + 0
        size: 35,35
        allow_stretch: True
    Label:
        text: "M.U.S.K"
        font_size:'40sp'
        bold: True
```

Fonte: Autoria Própria

Através da Figura 32 pode-se observar que a pseudo classe *ScreenOne* recebe os atributos tanto de *ScreenManager* quanto de *Button*. Podendo assim comportar-se tanto como uma tela variável, quanto ter características clicáveis de Touch. É importante notar

que todas as demais telas seguem este mesmo padrão de atributos, além também de partilharem a mesma cor de fundo e uma função clicável semelhante, essas por sua vez são definidas por:

```
background_color : 0,0,0,1
on_press: root.manager.current = 'screen two'
```

Quanto as características particulares desta classe temos o *Image* e o *Label*. Sendo que no *Image* determinou-se:

- **source:** Responsável por determinar a imagem utilizada, para tal é solicitado o seu diterório juntamente com o seu nome e a extensão do arquivo.
- **x, y:** Variáveis responsáveis por determinar a posição matricial inicial da imagem utilizada
- **size:** Apresenta as dimensões desejadas da imagem, permitindo assim o redimensionamento da imagem
- **allow stretch:** Flag, por padrão setada como False, permite o redimensionamento da imagem de acordo com o tamanho da tela e o size selecionado

A tela de sensoriamento segue um padrão bem semelhante a da tela anterior como pode ser observado a partir da Figura 33.

Figura 33 – Código Kivy - Interface (2)

```
<ScreenTwo@Button>:

    background_color: 0,0,0,1

    on_press:
        root.manager.current = 'screen three'

    Button:
        background_color: 0,0,0,1
        text: root.registrador_umid_e_temp
        size: 150,50
        font_size: 36
        pos_hint: {"center_x": .5, "center_y": .65}
```

Fonte: Autoria Própria

O código presente na Figura anterior gera então a seguinte tela.

Figura 34 – Kivy - Interface (2)



Fonte: Autoria Própria

A tela de dados internos por sua vez a mais diferente, pois, ao invés de ser construída em um arquivo kivy comum. Ela é gerada a partir de um arquivo python uma vez que esse tem melhores opções de recursividade. A código utilizado para gerar tal tela pode ser observado na Figura 35.

Figura 35 – Código Kivy - Interface (3)

Fonte: Autoria Própria

Semelhante ao código utilizado para gerar a tela de sensoriamento, a tela de dados internos também utiliza-se de botões para transmitir informações. Porém a geração destes botões se dá dentro de um `for` para que para cada informação dentro do banco de dados seja gerado um botão. Tal comportamento está apresentado na Figura 36.

Figura 36 – Kivy - Interface (3)



Fonte: Autoria Própria

4.6 Sistema de Segurança

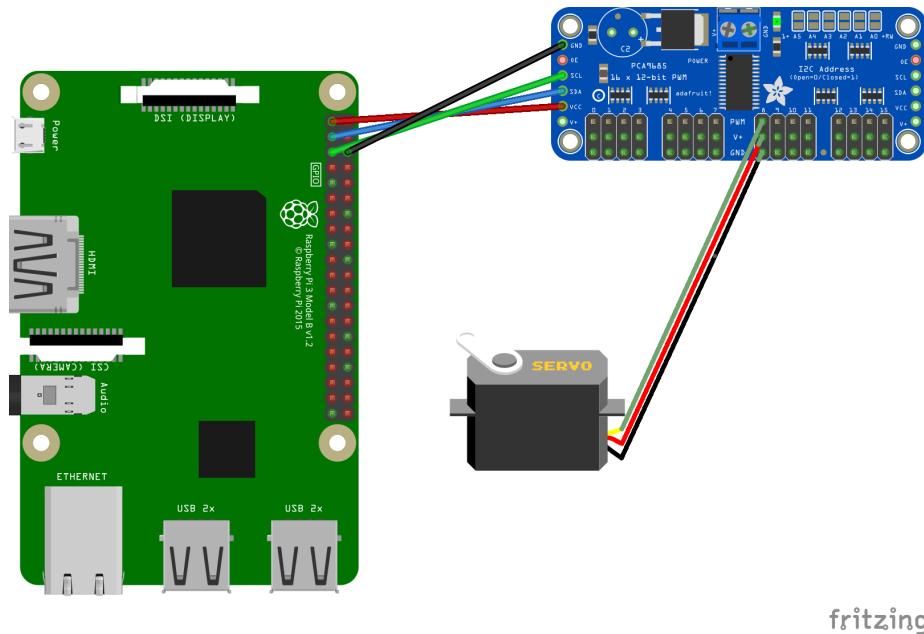
Uma das características principais do MUSK é a habilidade de atuar como um sistema de segurança no ambiente em que se localiza de forma ativa através da página Modo Vigia na Web e de serviços de email cadastrados pelo usuário. Para este protótipo foram desenvolvidas e validadas algumas formas de controle de segurança, com a possibilidade dessas serem expandidas de acordo com a necessidade do usuário.

4.6.1 Configuração do Motor

O motor possui 3 pinos de controle: VCC, GND e SIGNAL, esses são conectados na placa PCA9685 na GPIO 12. A placa é responsável por ser o driver de comunicação da RPi com o dispositivo eletromecânico. Já a conexão com a Raspberry é realizada nos pinos referentes ao protocolo de comunicação I2C, ou seja, as portas GPIO 2 e GPIO 3.

O esquemático de controle do servo está ilustrado na Figura 37 a seguir.

Figura 37 – Esquemático motores e RPi



Fonte: Autoria Própria

Em relação ao software, foi instalada a biblioteca *Adafruit_Python_PCA9685* para a integração do driver com a RPi. Os comandos executados no terminal para o processo de instalação estão descritos abaixo de forma sequencial.

```
sudo apt-get install git build-essential python-dev
sudo git clone https://github.com/adafruit/Adafruit_Python_PCA9685.git
cd Adafruit_Python_PCA9685
sudo python setup.py install
```

O controle da câmera é realizado pelo usuário a partir dos botões posicionados na página web do Modo Vigia que envia comandos ao servidor para ativar o funcionamento do motor deslocando-o em passos de 36° para a direção desejada. A comunicação com o backend é realizada a partir de um método de roteamento do framework Flask, conforme ilustrado na Figura 38 seguir.

Figura 38 – Rotas de controle do motor

```

@app.route('/panleft')
def move2():
    tccMotor.servo_position(0,-1)
    return redirect(url_for('camera.html'))

@app.route('/panright')
def move3():
    tccMotor.servo_position(0,1)
    return redirect(url_for('camera.html'))

```

Fonte: Autoria Própria

Assim a partir do botão que foi apertado, o programa identifica o motor que o usuário deseja movimentar e faz a chamada da função *servo_position*. Essa função pertence à classe *servoMotor*. Seus parâmetros de entrada são referentes ao canal do módulo PCA9685 em que o motor está conectado (canal 0) e ao tipo de movimento, onde 1 representa a direção horária e -1 a anti-horária. O retorno da requisição realiza um *refresh* da página *camera.html*.

Através do trecho de código anexado a baixo, pode-se entender melhor o funcionamento do controle do motor. O método *servo_position*, altera o atributo *memory*, que é responsável por guardar a posição atual do motor, de acordo com o sentido que se deseja girar o motor e em seguida gera o pulso que irá realizar o movimento do motor através do método *step*. Por fim é chamado o método *set_pwm* que executa o movimento do servo.

Figura 39 – Método *servo_position*

```

def servo_position(self, motor, pos):
    if pos == 1:
        self.memory[0]+=1
        print("pos = 1")
    if pos == -1:
        self.memory[0]-=1

    pulse = self.step(self.memory[0])
    self.pwm.set_pwm(0, 0, pulse)

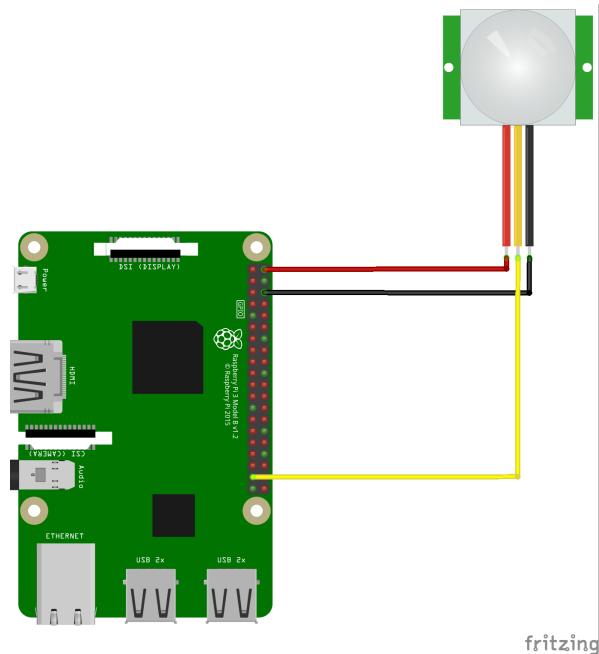
```

Fonte: Autoria Própria

4.6.2 Configuração do Sensor PIR

O sensor possui 3 pinos: *VCC*, *GND* e *OUTPUT*, esses são conectados na RPi conforme ilustra o esquemático na Figura 40 a seguir.

Figura 40 – Esquemático PIR



Fonte: Autoria Própria

O sensor foi configurado para atuar no modo de funcionamento H (*Auto-Reset*). A sensibilidade de detecção do dispositivo foi configurada para o seu valor máximo de forma a alcançar a maior área possível de verificação de presença. Já o tempo de saída foi definido para o seu valor mínimo de 3 segundos.

A partir da biblioteca *RPi.GPIO*, já instalada na raspberry, foi desenvolvido um código em python para testar algumas funcionalidades do sensor, como as variações de sensibilidade e tempo de saída, e em seguida foi implementado o código para o programa principal do projeto.

Para o controle eficiente das ações do sensor, foi criada a classe *sensor* que a partir do método *runConfiguration*, Figura 41, realiza a configuração dos pinos conectados ao sensor em modo de interrupção externa. A verificação do estado do dispositivo é realizada em uma nova *thread*, que ao receber um evento de interrupção (através de um sinal em subida de borda), realiza o acionamento da função de *callback*, chamada *motionCallback*, que é enviada como parâmetro de entrada do método de configuração da classe.

Figura 41 – Método de configuração do sensor PIR

```

def runConfiguration(self, motionFunction):
    GPIO.setmode(GPIO.BOARD)

    GPIO.setup(
        self.pinoMotion,
        GPIO.IN,
        pull_up_down=GPIO.PUD_DOWN
    )

    GPIO.add_event_detect(
        self.pinoMotion,
        GPIO.RISING,
        callback=motionFunction,
        bouncetime=300
    )

    print("[+]Sensors configured")

```

Fonte: Autoria Própria

A função de *callback*, por sua vez, retorna o método *sendEmail*, conforme a Figura 42 do objeto instanciado da classe *security* que realiza o processo de envio do email.

A classe *security* quando instanciada realiza a configuração dos parâmetros presentes no corpo do email. Para realizar a comunicação é necessário utilizar o protocolo *Simple Mail Transfer Protocol* (SMTP) que utiliza TCP/IP [62]. O atributo *server* é um objeto da classe desse protocolo, sendo assim responsável por todo o mecanismo de envio.

Assim, quando o método de enviar email é retornado pela função de *callback* este inicia a conexão com o servidor de email cadastrado (Google), realiza o login na conta de email do assistente e faz o envio da mensagem para a conta do usuário. A conexão tem um período curto e é encerrada no final de envio.

Figura 42 – Método de envio do email

```

def sendEmail(self):
    self.server.connect('smtp.gmail.com', '587')
    self.server.starttls()
    self.server.login(self.sender, self.password)
    self.server.sendmail(self.sender, self.receiver, self.message)
    self.server.quit()
    return print("[+]The email was send")

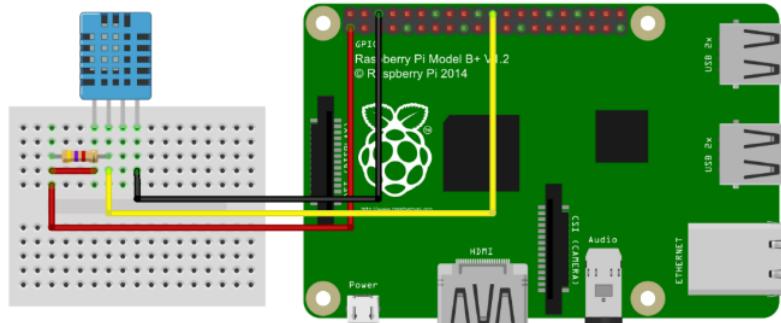
```

Fonte: Autoria Própria

4.6.3 Configuração DHT11

O sensor DHT11 foi configurado conforme ilustra a montagem da Figura 43 a seguir.

Figura 43 – Esquemático DHT11



Fonte: Autoria Própria

Os pinos da RPi utilizam um nível de tensão de 3,3V - as portas suportam 5V, mas a aplicação desse valor não é recomendado pelo fabricante. Assim é necessário alimentar o DHT11 com o pino de 3,3V da placa. Já para a leitura de dados é utilizado o pino 22 (GPIO 25), além de ser utilizado um resistor de pull-up de 4,7k na entrada da porta.

Para codificar o sensor é utilizado a biblioteca da Adafruit para python, instalando-a na RPi a partir dos comandos:

```
sudo git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install
```

Após as configurações no Raspbian, foi implementado um código de teste para o sensor a fim de verificar a confiabilidade do dispositivo. Para realizar a leitura do sensor DHT11, foi escrito o método *readDht11* dentro da classe *sensor*, conforme descrito na Figura 44 seguir:

Figura 44 – Método *readDht11*

```
def readDht11():

    umid, temp = Adafruit_DHT.read_retry(
        self.sensorTemperatureType,
        self.pinoTemperature
    )

    return umid, temp
```

Fonte: Autoria Própria

O método *read_retry* referente a biblioteca da *Adafruit* para o sensor informa os valores da umidade e temperatura lidas pelo sensor. Esses valores são então retornados pelo método *readDht11* como resposta a requisição realizada pelo usuário através do lado

do cliente. A rota de chamada definida utilizando o framework Flask está exemplificado na Figura 45.

Figura 45 – Rota para receber os valores do sensor DHT11

```
@app.route('/temperatura')
def temp():

    umidade, temperatura = sensor.readDht11()
    templateData = {'umid': umidade, 'temp': temperatura}

    return render_template('temperaturahtml.html', **templateData)
```

Fonte: Autoria Própria

Em seguida os dados recebidos são convertidos para um template padrão e retornados junto com um *refresh* da página *temperatura.html*.

4.6.4 Configuração OpenCV

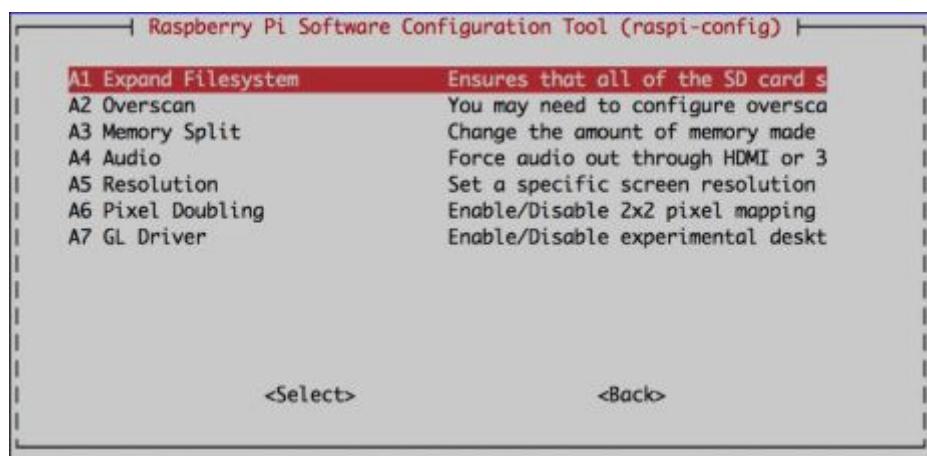
A instalação do OpenCV foi realizada a partir da compilação do seu código fonte, uma vez que as tentativas de instalação via instalador de pacotes *pip* não foram frutíferas. Observou-se, também, que devido ao elevado tamanho do OpenCV seria necessário o aumento do memória de armazenamento, desta forma, refez-se todas as configurações e alterações anteriores sobre um cartão SD de 32 Gigabytes.

A instalação do OpenCV iniciou-se com a expansão da memória utilizada pelo sistema, pra que todo o espaço disponível fosse utilizado. Para isso a partir do terminal, utilizou-se o comando:

```
sudo raspi-config
```

Selecionou-se "Advanced Options", conforme ilustra na Figura 46.

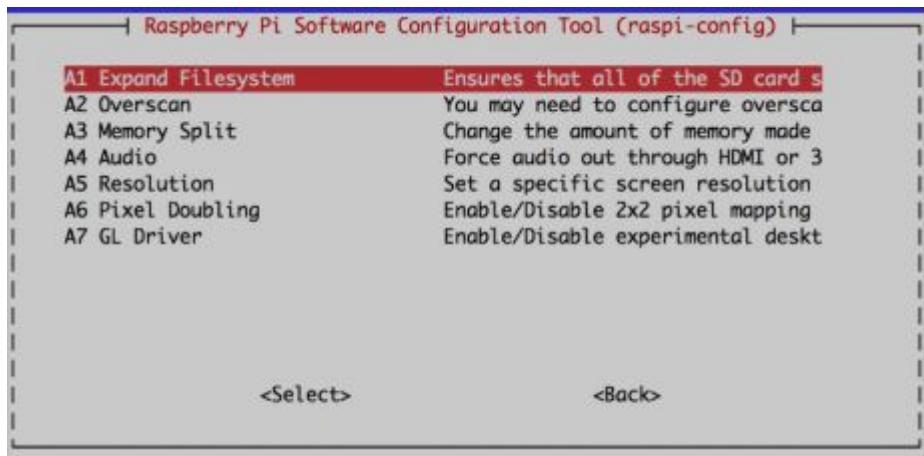
Figura 46 – Menu de Configuração RPi, Opções Avançadas



Fonte: Autoria Própria

Em seguida *"Expand Filesystem"*, conforme apresentado na Figura 47.

Figura 47 – Menu de Configuração RPi, Expandir Sistema



Fonte: Autoria Própria

Para que as alterações sejam postas em prática reiniciou-se o sistema com o comando:

```
sudo reboot now
```

Com espaço o suficiente para a instalação faz-se necessário a atualização dos pacotes do sistema operacional. Realizada a partir dos comandos:

```
sudo apt-get update
sudo apt-get upgrade
```

Segue-se com a instalação dos pacotes de desenvolvimento:

```
sudo apt-get install build-essential cmake unzip pkg-config
```

Deve-se então instalar as bibliotecas para o controle de imagem e vídeo.

```
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev lib
sudo apt-get install libxvidcore-dev libx264-dev
```

Para que as funções de HMI (Human-Machine Interface) sejam implementadas é necessário a instalação das bibliotecas de GTK e GUI (Grafical User Interface).

```
sudo apt-get install libgtk-3-dev
sudo apt-get install libcanberra-gtk*
```

O asterisco presente na função anterior tem como propósito a instalação da biblioteca específica para os processadores ARM (presente na RPi).

Para um melhor desempenho instalou-se também os pacotes de otimizações numéricas para OpenCV juntamente com alguns *headers* de desenvolvimento.

```
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get install python3-dev
sudo pip install numpy
```

Depois de realizar todas as preparações iniciou-se a instalação do OpenCV propriamente dito. Para isso é necessário fazer o download dos repositórios.

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.0.0.zip
```

É necessário a descompactação dos arquivos e para maior facilidade a renomeação dos seus diretórios.

```
unzip opencv.zip
unzip opencv_contrib.zip
mv opencv-4.0.0 opencv
mv opencv_contrib-4.0.0 opencv_contrib
```

Com o código fonte devidamente presente na RPi, o próximo passo é a sua compilação. Para isso deve-se entrar na pasta descompactada e criar um diretório, o qual por padrão foi chamado de *build*. Pode-se observar esse procedimento nos comandos:

```
cd /opencv
mkdir build
cd build
```

Para preparar o diretório para a compilação do código utilizou-se o comando *cmake* com as seguintes configurações:

```
cmake -D CMAKE_BUILD_TYPE = RELEASE \
-D CMAKE_INSTALL_PREFIX = /usr/local \
-D OPENCV_EXTRA_MODULES_PATH = /opencv_contrib/modules \
-D ENABLE_NEON = ON \
-D ENABLE_VFPV3 = ON \
```

```
-D BUILD_TESTS = OFF \
-D OPENCV_ENABLE_NONFREE = ON \
-D INSTALL_PYTHON_EXAMPLES = OFF \
-D BUILD_EXAMPLES = OFF
```

Ao fim desta operação foi exibido um *log* de informações indicando o resultado positivo da compilação.

Para continuar com a instalação do OpenCV foi necessário aumentar o espaço definido para a memória virtual, ou como é mais comumente chamada *swap space*. Para isso seguiu-se os seguintes passos:

1. Abriu-se o arquivo responsável por determinar o espaço da memória virtual:

```
sudo nano /etc/dphys-swapfile
```

2. Editou-se o arquivo de forma a deixar a memória anterior comentada e fazer com que esta passe a ser de 2 GB ao invés dos 100 MB anteriores.

```
# CONF_SWAPSIZE = 100
CONF_SWAPSIZE = 2048
```

3. Reiniciou-se o espaço virtual da RPi com os comandos:

```
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

Compilou-se o código fonte do OpenCV com o comando:

```
make -j4
```

Este comando então gerou o seguinte *log* de saída sobre o terminal indicado os resultados obtidos:

Por fim, instalou-se o OpenCV e retornou-se com os comandos:

```
sudo make install
sudo ldconfig
```

4.6.4.1 Preparando Ambiente OpenCV

Devido a sua característica modular além da instalação do OpenCV é necessário também algumas pequenas adaptações para que seja possível utilizar essa biblioteca em diferentes sistemas.

Para tal primeiramente verificou-se se a instalação do OpenCV ocorreu sem erros. A verificação se deu a partir da sequência de comandos.

```
sudo python3
>> import cv2
>> cv2.__version__
```

Após tal verificação, tornou-se possível a utilização de comandos de imagem básicos. Para que comandos mais complicados de detecção facial sejam utilizados fez-se necessário importar os módulos correspondentes para o diretório de desenvolvimento. Para que isto fosse realizado, era preciso determinar-se o diretório de armazenamento desses módulos, seguiu-se então os comandos.

```
sudo python3
>> import cv2
>> print(cv2.__file__)
```

O resultado obtido foi o o caminho de diretório:

/usr/lib/python3.4/ - Colocar aqui o diretório

Dado este caminho, decidiu-se pela sua cópia em um diretório específico para o armazenamento de módulos do OpenCV. Este diretório foi chamado de:

/Desktop/tcc_files/opencv_aws/cascade/data.

O módulo copiado para o diretório citado anteriormente foi o *frontal face cascade*, o qual tem como principal função a detecção de frontal de rostos.

Para a criação do diretório e a cópia do modulo em questão para o diretório desejado seguiu-se os passos:

```
sudo mkdir /Desktop/tcc_files/opencv_aws/cascade/data
cd /Desktop/tcc_files/opencv_aws/cascade/data
sudo cp -rf /usr/lib/python3.4/data /Desktop/tcc_files/opencv_aws/cascade/data
```

Para conferir que este processo obteve sucesso, gerou-se e compilou-se o seguinte código em python, apresentado na Figura 48.

Figura 48 – Método teste OpenCV

```
#!/usr/bin/python3
# -*- coding: utf-8

import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('cascade/data/haarcascade_frontalface_alt2.xml')
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
```

Fonte: Autoria Própria

4.6.4.2 Implementação Geral

Para a implementação do código do OpenCV criou-se uma classe em um arquivo python. Onde tanto a classe quanto o arquivo receberam o nome de *camera*. Decidiu-se por utilizar-se deste método primeiramente pelo rigor organizacional do projeto como um todo, além da flexibilidade de sua utilização em diferentes áreas do projeto.

A classe *camera* conta com seis métodos. Sendo que dentre estes os mais importantes para o entendimento do funcionamento do programa são o *getImage*, *runFaceDetector* e *sendVideo*.

O método *runFaceDetector* é normalmente chamado por programas exteriores e funciona como uma função de alto nível lidando com outros métodos de mais baixo nível como pode ser observado a partir da imagem 49.

Figura 49 – Método *runFaceDetector*

```
def runFaceDetector(self):
    self.flag = False
    ret, self.frame = self.getFrame();

    #turn into gray
    self.gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    self.faces = self.faceCascade.detectMultiScale(
        self.gray_frame,
        scaleFactor=1.5,
        minNeighbors=8
    )
```

Fonte: Autoria Própria

Inicialmente determina-se uma *flag* que indica que ainda não se notou um rosto.

Em seguida chama-se o método *getFrame* que tem como função capturar as imagens adquiridas pela câmera, converte-se essa imagem para preto e branco. Através dessa nova imagem compara-se, com o comando *detectMultiScale*, a existência de olhos, boca e nariz para que dai sejam retornado um vetor com valores de onde este rosto está localizado na imagem.

Após isso é chamado o método *getImage*, o qual pode ser compreendido através da Figura 50.

Figura 50 – Método *getImage*

```
def getImage(self):
    for (x,y,w,h) in self.faces:
        print(x,y,w,h)
        #Cords
        end_cord_x = x+w+50; end_cord_y = y+h+50;
        ini_cord_x = x-25; ini_cord_y = y-25;
        #roi - region of interest
        roi_gray = self.gray_frame[ini_cord_y:end_cord_y, ini_cord_x:end_cord_x]
        roi_color = self.frame[ini_cord_y:end_cord_y, ini_cord_x:end_cord_x]
        roi_gray_res = cv2.resize(roi_gray, dsize=(420, 420), interpolation=cv2.INTER_CUBIC)
        cv2.imwrite("img1.jpg", roi_gray_res)
        cv2.rectangle(self.frame, (x,y), (end_cord_x,end_cord_y), self.colorRect, self.lineRect)
    self.flag = True
```

Fonte: Autoria Própria

A principal função desta definição é delimitar a região de interesse. Realizar as operações de redimensionar as imagens e salva-la para que essa então fosse possível ser comparada com as coleções armazenadas na AWS.

4.6.4.3 Implementação WEB

Por fim, a definição *sendVideo*, Figura 51 tem como objetivo lidar com a geração de imagens para a apresentação via WEB.

Figura 51 – Método *sendVideo*

```
def sendVideo(self):
    success, image = self.getFrame()
    ret, jpg = cv2.imencode('.jpg', image)
    return jpg.tobytes()
```

Fonte: Autoria Própria

O método *sendVideo* é solicitado pela definição *generateVideo* presente no arquivo *server.py*. O qual é responsável por lidar com todas as solicitações envolvidas na parte WEB. A Figura 52 apresenta a definição *generateVideo*, a partir da qual pode-se verificar que a mesma quando chamada utiliza-se de uma função em *loop* para primeiramente

adquirir a imagem da câmera e após isso remontá-la e enviá-la para página WEB como uma imagem JPEG. Desta forma, devido ao loop a sequência de imagens assemelha-se a geração de um vídeo em tempo real.

Figura 52 – Método *generateVideo*

```
def generateVideo(cam):
    while True:
        frame = cam.sendVideo()
        yield (b"--frame\r\n"
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
```

Fonte: Autoria Própria

Por último a definição *generateVideo* é chamada quando o usuário acessa a página WEB correspondente. Sendo o endereço (ou URL ou rota) desta página, /cameraview. Para que a imagem obtida na câmera seja apresentada na página WEB retorna-se a imagem obtida através da função *generateVideo* com os devidos cuidados como apresentado na Figura 53.

Figura 53 – Endereço *cameraview*

```
@app.route('/cameraview')
def cameraview():
    return Response(
        generateVideo(camera()),
        mimetype='multipart/x-mixed-replace; boundary=frame'
    )
```

Fonte: Autoria Própria

4.6.5 Configuração AWS Rekognition

Para a utilização dos serviços da AWS é necessário realizar o cadastro no site oficial, assim sua conta é automaticamente cadastrada em todos os serviços disponíveis. Apesar de ser um serviço pago, a Amazon oferece o uso grátis por um ano de forma limitada. No escopo desse projeto foi integrado a API do Rekognition, que na versão gratuita tem as seguintes características [63]:

- Analise de 5000 imagens por mês;
- Armazenamento de 1000 metadados de faces por mês;

Além disso, em todos os serviços é pago apenas o que foi utilizado, conforme especificado no site.

- Vídeo arquivado analisado (faturado por segundo): 0,10 USD por minuto;
- Vídeo de streaming ao vivo analisado (faturado por segundo): 0,12 USD por minuto;
- Preço por 1000 metadados faciais armazenados: 0,01 USD por mês;

Após o cadastro, é necessário criar um usuário do *Identity and Access Management* (IAM) com as permissões de administrador. Este usuário é necessário para definir quais serviços serão acessíveis pelo usuário, uma vez que a AWS exige a apresentação de credenciais para chamar as APIs. Assim, foi realizado os seguintes passos [64]:

- Criação do usuário *rasptcc*;
- Anexo das permissões *AdministratorAccess* e *AmazonRekognitionFullAccess*.

Para a integração do projeto com o Rekognition, foi utilizada a biblioteca *boto3*, que é a SDK da AWS para python e utiliza o protocolo de comunicação HTTP. O processo de instalação é realizado a partir do comando abaixo [65]:

```
sudo pip3 install boto3
```

Após configurar o ambiente de desenvolvimento, foi necessário efetuar as etapas de tratamento das imagens armazenadas e enviadas, criação das coleções e treino dos modelos para definição dos vetores de características.

O tratamento das imagens para o treinamento dos modelos e criação dos vetores de características faciais, que serão utilizados para o reconhecimento facial, seguem alguns pontos sugeridos pela Amazon que estão sintetizadas a seguir [66]:

- Utilização de imagens com os olhos abertos e visíveis;
- Imagens em que as faces tenham um tamanho mínimo de 50 x 50 pixels e máxima de 1920 x 1080 pixels;
- Utilização de imagens coloridas;
- Utilização de imagens de uma mesma pessoa de diversos ângulos (30° , 45° e 90°) e distâncias (0.5m, 0.75m e 1m);
- Imagens que não estejam desfocadas.

Em relação ao tamanho das imagens, foram definidos dois formatos, as imagens de treino do modelo tinham 780 x 1040 pixels devido ao fato desse ser o tamanho original de

captura; Já as imagens de busca, ou seja, de verificação da identidade da pessoa, possuem o tamanho de 400 x 650 pixels de forma a otimizar o processo, pois proporção influencia diretamente no tempo de processamento na nuvem.

A criação da coleção é executada através da função *create_collection* que recebe como parâmetro de entrada o nome da coleção utilizada no trabalho *collectionTcc* [67].

Figura 54 – Função *create_collection*

```
response = client.create_collection(
    CollectionId='string'
)
```

Fonte: Autoria Própria

As imagens para treino são enviadas a partir da chamada da funcionalidade *index_faces* da API junto à definição dos seus parâmetros de entrada [67].

Figura 55 – Função *index_face*

```
response = client.index_faces(
    CollectionId='string',
    Image={
        'Bytes': b'bytes',
        'S3Object': {
            'Bucket': 'string',
            'Name': 'string',
            'Version': 'string'
        }
    },
    ExternalImageId='string',
    DetectionAttributes=[
        'DEFAULT' | 'ALL',
    ],
    MaxFaces=123,
    QualityFilter='NONE' | 'AUTO'
)
```

Fonte: Autoria Própria

Para o parâmetro de entrada *Image* foi passado os bytes da imagem que estava armazenada na própria RPi. Para o *ExternalImageId*, que é o rótulo da imagem, foi definido um padrão de nomenclatura utilizando o primeiro e segundo nome da pessoa com letras minusculas. A *CollectionId* recebe o nome da coleção criada para o desenvolvimento do projeto.

A função utilizada para realizar a identificação facial está ilustrada na Figura 56, seguida de uma análise da lógica implementada.

Figura 56 – Função *faceMatch*

```

def faceMatch(self, client):
    result, resp = self.checkFace(client);
    if(result):
        resu, res = self.checkMatches(client);
        if resu:
            self.name = res['FaceMatches'][0]['Face']['ExternalImageId']
            self.similarity = res['FaceMatches'][0]['Similarity']
            self.confidence = res['FaceMatches'][0]['Face']['Confidence']
        else :
            self.name = 'unknown'
            self.similarity = 0
            self.confidence = 0
        print("[+]Detected Person: " + str(self.name))
        print("[+]Similarity: " + str(self.similarity))
        print("[+]Confidence: " + str(self.confidence))
        print(" ")
    else:
        print("[+]No faces detected")

```

Fonte: Autoria Própria

Com o intuito de garantir a confiabilidade do serviço de reconhecimento facial, é realizada uma segunda confirmação da existência de faces no ambiente através do *Rekognition*. Ou seja, após a foto ser tirada, esta é analisada através da chamada da função *checkFace*. O resultado é armazenado na variável *result*, que caso satisfaça a condição dentro do *if*, ou seja, se for verdadeiro, inicia-se o processo de identificação facial através da função *checkMatches*.

Após o reconhecimento facial, o *Rekognition* retorna uma lista de informações sobre a análise realizada. Essas informações são imprimidas no prompt de comando e em seguida armazenadas em arquivo de log.

4.7 Comando de Voz

4.7.1 Configuração do Google Assistant

Apesar da imagem do Raspbian já possuir todas as dependências instaladas do Assistant, ainda foi necessário definir algumas configurações para os periféricos e para a comunicação da raspberry com a Google Cloud. O processo de instalação seguiu o descrito no tutorial do site *Android Authority* [28].

O primeiro passo foi a configuração da porta de áudio. No *Start dev terminal* foi efetuado o comando:

```
sudo leafpad /boot/config.txt
```

Assim, foi comentada as linhas ilustradas na Figura 57.

Figura 57 – Configuração do arquivo config.txt

```
# Enable audio (loads snd_bcm2835)
dtoverlay=audio=on
#dtoverlay=i2s-mmap
#dtoverlay=googlevoicehat-soundcard
```

Fonte: Autoria Própria

Para utilizar o speaker e o microfone é preciso indicar para a RPi quais são as portas correspondentes. O arquivo *asounf.conf* é o responsável por tal configuração. Assim, foi executado o comando no cmd.

```
sudo leafpad /etc/asound.conf
```

O arquivo foi atualizado da seguinte forma.

Figura 58 – Configuração do arquivo *asounf.conf*

```
pcm.!default {
    type asym
    capture.pcm "mic"
    playback.pcm "speaker"
}
pcm.mic {
    type plug
    slave {
        pcm "hw:1,0"
    }
}
pcm.speaker {
    type plug
    slave {
        pcm "hw:0,0"
    }
}
```

Fonte: Autoria Própria

Em seguida executou-se no *Start dev terminal* comando:

```
sudo leafpad /home/pi/voice-recognizer-raspi/checkpoints/check_audio.py
```

Para utilizar o microfone usb é necessário alterar a variável *VOICEHAT_ID* = ‘*googlevoicehat*’ para *VOICEHAT_ID* = ‘*bcm2835*’, uma vez que o ambiente está configurado para utilizar o microfone padrão do kit de desenvolvimento do Google.

Para verificar se os periféricos estão sendo reconhecidos pela interface, executou-se o script bash abaixo.

```
sudo bash check_audio
```

Após isso é necessário configurar o Google Cloud, através de uma conta do próprio Google para realizar a autenticação do dispositivo a partir das credenciais de acesso. O procedimento dessa etapa está descrita a seguir.

- Criar um novo projeto;
- Habilitar o *Google Assistant API*;
- Criar as credenciais de *OAuth client ID*;
- Executar o download do arquivo da credencial;
- Alterar o nome da credencial para *assistant.json*;
- Mover o arquivo para a pasta */home/pi/assistant.json*;
- Habilitar no painel de Controle de Atividades: atividade na Web e de apps, histórico de localização, informações do dispositivo e atividade de áudio e voz.

Em seguida foi executado através do *Start dev terminal* o comando a baixo para verificar se as configurações do sistema estavam funcionando corretamente a partir de comandos de voz.

```
./assistant_grpc_demo.py
```

Após a conclusão dessas etapas de configuração do *Google Cloud*, iniciou-se o desenvolvimento de novas *skills* para a realização de tarefas específicas do assistente pessoal.

4.7.2 Desenvolvimento de Skills

Para o desenvolvimento não foi utilizado o método convencional definido pelo Google, devido a problemas de desatualização da versão do SO instalado. Assim, as skills foram criadas através de uma adaptação do arquivo *python assistant_grpc_demo.py*. Para auxiliar nessa etapa foi necessário instalar o *Google Text-to-Speech (gtts)*, que é uma biblioteca em *python* do google que transforma textos em áudio. O processo de instalação foi simples e utilizando o *pip*.

```
pip install gTTS
```

Em seguida, foi definido o método *speak*, conforme ilustrado na Figura 59 seguir.

Figura 59 – Método *speak*

```

def speak(text_rec):
    """Text to speech"""
    try:
        from gtts import gTTS
        from playsound import playsound

        tts = gTTS(text=text_rec, lang='en')
        tts.save("good.mp3")
        playsound('good.mp3')

    except Exception:
        raise

```

Fonte: Autoria Própria

Quando o método é chamado, ele requisita o parâmetro de entrada *text_rec* que é o texto reconhecido pelo serviço do *google assistant*. Em seguida, ele faz a chamada da função *gTTS* que converte o texto de entrada para áudio. O áudio é então executado a partir da biblioteca *playsound*.

Após essa etapa de preparação, foi criado uma nova skill que informa as tarefas que o usuário tem agendada. O desenvolvimento está ilustrado na Figura 60.

Figura 60 – Desenvolvimento de skills

```

text, audio = assistant.recognize()
if text is not None:
    if text == 'what do I have today':
        tarefas = actionTask()
        speak(tarefas);
        audio = None

    if text == 'turn off the lights':
        speak('turnning off the lights');
        audio = None

    if text == 'goodbye':
        status_ui.status('stopping')
        print('Bye!')
        break

if audio is not None:
    aiy.audio.play_audio(audio)

```

Fonte: Autoria Própria

As novas skills não estão no banco de dados do Google, assim uma forma de verificar se estas foram requisitas, é verificar se o texto convertido pelo Google é equivalente a alguma skill desenvolvida. A análise é realizada através do condicional *if*. O processo completo é descrito a seguir.

Após executar o serviço do Google, ele aguarda um gatilho, que é o inicio da fala do usuário. Após a interação homem-maquina o serviço faz o reconhecimento do texto falado e armazena os resultados obtidos nas variáveis *text* e *audio*, através da função *assistant.recognize*. Dentro das variáveis são armazenados, respectivamente, o texto convertido e a resposta em áudio gerada pelo Google. No entanto, caso o *assistant* não encontre uma resposta pré-configurada no seu banco, ele retorna um áudio padrão informando que não foi possível realizar a tarefa solicitada.

Em seguida, é realizada a verificação, através dos condicionais *if*, se o comando passado pelo usuário corresponde a skill de tarefas criada. Quando sua condição é satisfeita, o programa consulta as tarefas pendentes do usuário através da chamada do método *actionTask* e em seguida utiliza o *gTTS* para realizar a interação via áudio.

4.8 Desenvolvimento do Banco de Dados

O banco de dados mostra-se fundamental pois será a partir dele que as instâncias web podem não apenas salvar e consultar dados mas também podem se comunicar com os elementos locais tal como a interface gráfica.

A criação do banco de dados ocorreu a partir do SQLite (sqlite3) e do SQLALCHEMY (Flask Alchemy) – presente no código em python. A criação do banco de dados pode ocorrer tanto diretamente a partir do console quanto a partir de código presente em um arquivo python. Ao longo deste trabalho e em diferentes programas foram-se utilizado os dois métodos a depender da complexidade e da necessidade do programa em questão.

Para criar um banco de dados via terminal deve-se primeiramente iniciá-lo através do comando a seguir

```
sqlite3 <nome>.db
```

A seguir pode-se abri-lo e conferir que ainda não existe nenhuma Tabela dentro do mesmo.

```
.table
```

Após isso deve-se fechar a base de dados e rodar o programa e instanciar a base de dados a partir deste arquivo, para tal deve-se digitar o comando:

```
from <arquivopython>.py import <nome da Tabela do banco de dados>
```

Deve-se então criar a Tabela do banco de dados instanciado.

```
<nome da Tabela do banco de dados>.create_all() exit()
```

Agora se iniciar novamente o banco de dados e buscar por suas Tabelas deverá se achar a Tabela “<nome da Tabela do banco de dados>”.

Na Figura 61 seguir está presente o arquivo python que permite instanciar a Tabela.

Figura 61 – Banco de dados python (1)

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///home/pi/web-server/todo.db'

db = SQLAlchemy(app)

class Todo(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String(200))
    complete = db.Column(db.Boolean)
```

Fonte: Autoria Própria

O segundo método de se iniciar uma Tabela no banco de dados pode ser observada no código da Figura 62 abaixo a partir da chamada da função *create_table*.

Figura 62 – Banco de dados python (2)

```
import os
import sqlite3

conn = sqlite3.connect('todo.db') # cria se necessário a base de dados
c = conn.cursor() # gera o cursor da db

# gera se necessário a tabela do db
def create_table():
    c.execute("CREATE TABLE IF NOT EXISTS Todo(id INTEGER, title TEXT, start DATETIME, complete NUMERIC)")
```

Fonte: Autoria Própria

4.9 Característica Open-Source

Para que o projeto seja acessível para toda a comunidade interessada utilizou-se a ferramenta e ambiente GitHub. A criação do ambiente de armazenamento *open-source* do projeto deu-se a partir da interface web do GitHub, onde através desta gerou-se primeiramente um usuário administrador(neste caso o estudante Samuel Santos) do projeto e em seguida os diretórios necessários para o armazenamento do projeto e suas características básicas.

O acesso a tais arquivos se dá através do link presente a seguir e pode ser acessado por todos usuários. Além de que qualquer membro da comunidade GitHub pode solicitar

o poder de alteração do código, a qual pode ser realizada mediante a autorização do administrador.

Link de acesso: <https://github.com/samuelsantos22/M.U.S.K.git>

5 Resultados e Discussões

Neste capítulo foi apresentado todos os resultados qualitativos e quantitativos da interação com o usuário, do sistema de segurança e do desempenho do sistema. Em paralelo, foi discutido os indicadores obtidos, pontuando possíveis aspectos de melhora no projeto. No final é destinada uma seção para apresentar futuros aprimoramentos do assistente pessoal.

5.1 Interação com o Usuário

Um dos elementos fundamentais para todo assistente pessoal é a sua capacidade de interação com os seus usuários. Para uma interação mais completa e integrada buscou-se que a mesma fosse capaz de fornecer todo o seu potencial através de telas e comandos simples.

A interação entre o sistema e o usuário se dá principalmente de forma visual, sendo que parte das funções pode ser acessada através da tela do componente e outra parte a partir da interface WEB. A escolha de tal separação se deu para que a utilização do sistema ocorresse da forma mais orgânica possível, uma vez que não se apresenta uma solução viável a configuração do dispositivo através de uma tela móvel como a presente no dispositivo.

5.1.1 Web

5.1.1.1 Página Inicial

A interface web inicia-se com uma tela de entrada na qual é apresentado o assistente pessoal – MUSK -, além de suas principais funções e as guias que levam as demais funcionalidades. A Figura 63 apresenta a página de entrada.

Figura 63 – Página de Entrada WEB



Fonte: Autoria Própria

5.1.1.2 Página de Aplicativos

Ao acessar o site tem-se a pagina interna a qual permite que se escolha e adicione aplicativos web, além é claro da descrição de cada um dos aplicativos. A Figura 64 apresenta a home de entrada.

Figura 64 – Página de Aplicativos

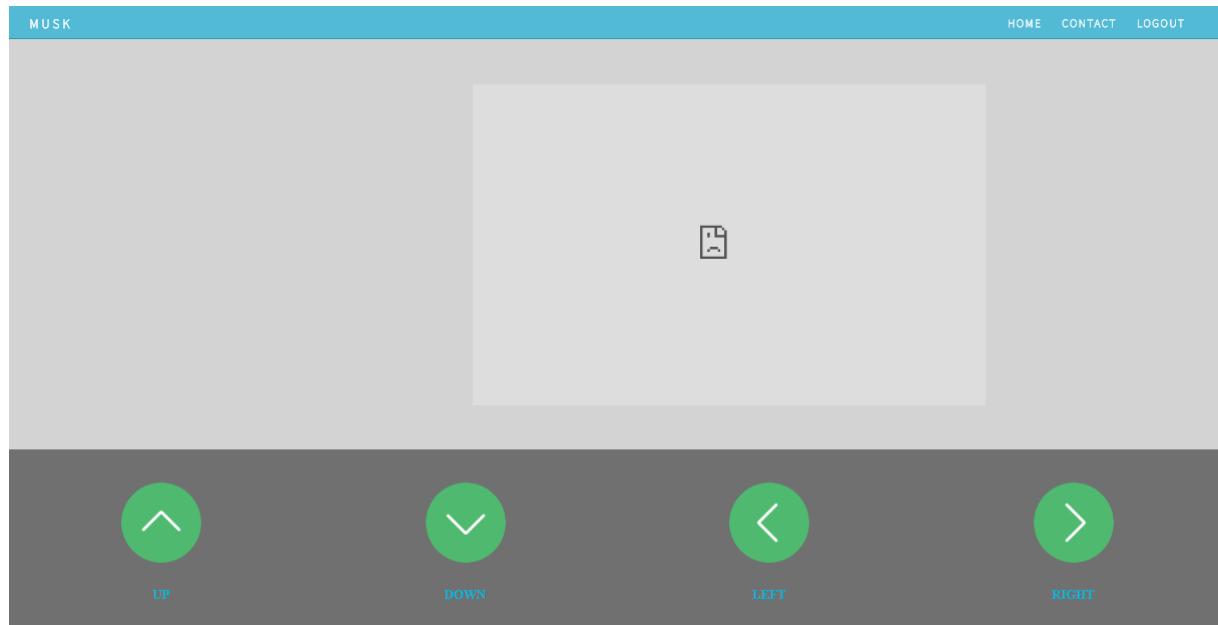


Fonte: Autoria Própria

5.1.1.3 Página Modo Vigia

O modo vigia permite que se observe a partir de qualquer lugar através do site as imagens obtidas a partir da câmera usb. Além de que permitir também que se altere a posição da câmera aumentando assim consideravelmente o ângulo de visão da mesma. A Figura 65 apresenta um exemplo da interface desta função web.

Figura 65 – Exemplo Interface Modo Vigia

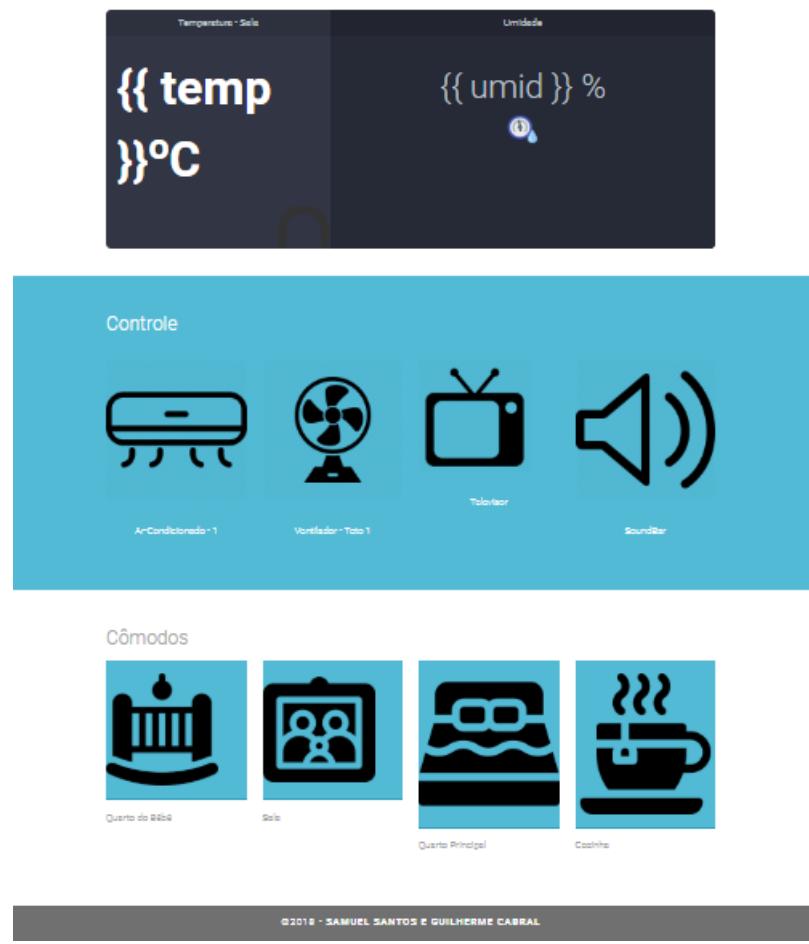


Fonte: Autoria Própria

5.1.1.4 Sensor de Temperatura e Umidade

A interface do sensor de umidade e temperatura está presente de forma simples em um exemplo na Figura 66. Através dele é possível verificar localmente o ambiente e futuramente será possível até mesmo acionar determinados eletrodomésticos para controlar esta temperatura e torná-la ideal ao indicado pelo usuário.

Figura 66 – Interface de Temperatura e Umidade

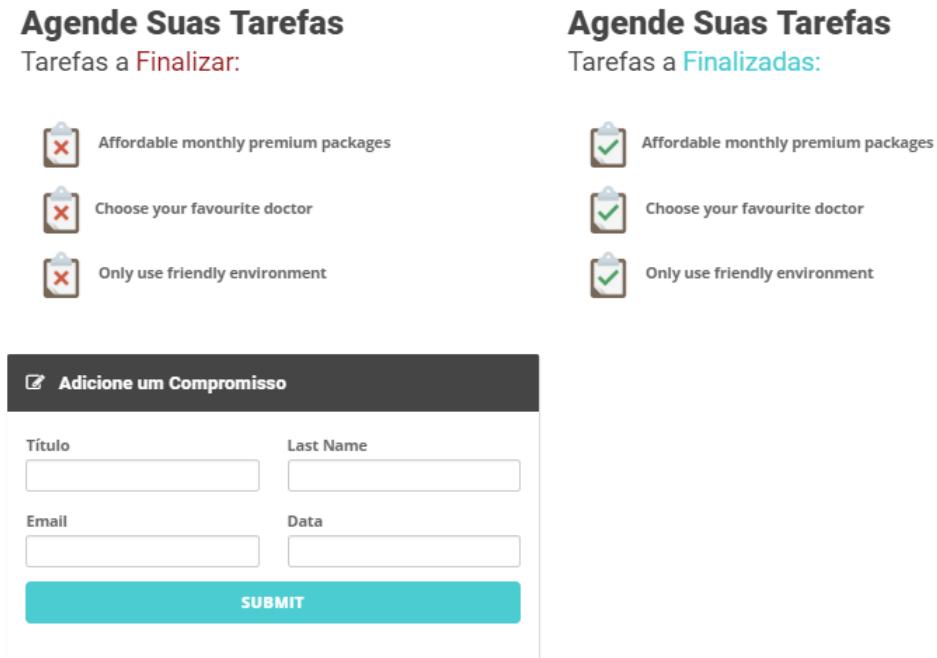


Fonte: Autoria Própria

5.1.1.5 Agenda WEB

Por fim tem-se a agenda que indica os eventos futuros, os quais por enquanto só podem ser adicionados através do site mas são mostrados todos os dias na tela do assistente pessoal.

Figura 67 – Interface Agenda



Fonte: Autoria Própria

5.2 Sistema de Segurança

A capacidade do assistente realizar um serviço de segurança do ambiente, mostrou-se muito promissora devido aos resultados obtidos nos testes realizados. Além disso foi demonstrando a possibilidade de expansão do sistema com a integração de mais dispositivos. A seguir, está apresentado os resultados obtidos para cada componente que compõe o sistema de segurança.

5.2.1 Controle Câmera

Para a interação do usuário com a câmera, efetuou-se a validação do controle de um módulo físico através de comandos realizados pelo usuário do lado do cliente no web server. O processo obteve resultados com um tempo de resposta com *delay* mínimo e não perceptivo. Além disso, pode-se garantir a partir dessa funcionalidade a possibilidade de integrar mais periféricos com o MUSK com conexão direta com os pinos do hardware ou através da conexão via Wi-Fi.

Um outro ponto a ser ressaltado é o fato de que toda a integração foi construída dentro de uma mesma rede, ou seja, caso o usuário acesse a página web de uma rede externa pode ocorrer variações no resultado devido ao desempenho e qualidade do sinal de internet.

A dificuldade encontrada na integração dessa etapa no projeto foi devido a defini-

ção do número de graus de liberdade de controle da câmera. O assistente permite apenas um deslocamento de um eixo lateral, o que reflete no movimento da câmera. Na fase de pré-projeto foi idealizado um deslocamento com dois graus de liberdade, no entanto devido ao peso do protótipo não foi possível utilizar um suporte pan-tilt, que possibilitaria uma maior mobilidade para a câmera integrada ao MUSK.

5.2.2 Detector de Presença

Outro módulo acoplado ao protótipo é o sensor de presença PIR. Seu objetivo é garantir a segurança do ambiente quando o usuário não está presente, assim a sua ação só é realizada quando for ativada via interface Web. Quando o sensor detecta uma nova presença no ambiente é enviado uma mensagem para o email do responsável cadastrado.

A detecção de uma nova presença no ambiente teve resultados excelentes, apresentando um delay desprezível (da ordem de milisegundos) e uma sensibilidade alta. Em relação ao processo do email, obteve-se uma latência baixa, ou seja, quase instantânea de entrega da mensagem, comprovando a eficiência dos métodos e protocolos utilizados na operação.

A dificuldade encontrada na integração do sensor com o sistema é relacionada a conexão via cabo entre as duas partes. Esse fator limita o alcance do PIR, uma vez que é recomendado instalar o sensor em lugares altos, acima de 2 metros, para obter uma "visão" ampla do ambiente. Assim, uma possível solução para esse problema é a utilização de um microcontrolador para controlar o sensor PIR e comunicar os eventos obtidos para a RPi via internet. Essa solução permitiria ainda a integração de diversos sensores com o sistema, monitorando assim quantos ambientes o usuário desejar.

5.2.3 Reconhecimento Facial

Um outro componente presente no serviço de segurança disponibilizado pelo MUSK é capacidade de realizar a detecção e reconhecimento de faces. Isso permite que o protótipo possa monitorar as pessoas que estão presentes no ambiente, além de controlar as permissões e acesso a uma casa ou escritório. Por exemplo, a partir do cadastro de pessoas conhecidas, o assistente é capaz de liberar a entrada de uma pessoa na empresa de forma inteligente, rápida e gerando logs de acesso. Para validar a eficiência dessa habilidade, foram realizados alguns testes de performance do reconhecimento de faces.

5.2.3.1 Testes AWS Rekognition

Para verificar a confiabilidade e latência da API da Amazon, foi desenvolvido *scripts* em python para extrair informações que contribuam na avaliação do serviço. Foram realizados 2 processos:

- Processo 1: Análise de fotos pré-definidas;
- Processo 2: Análise de imagens geradas em tempo real.

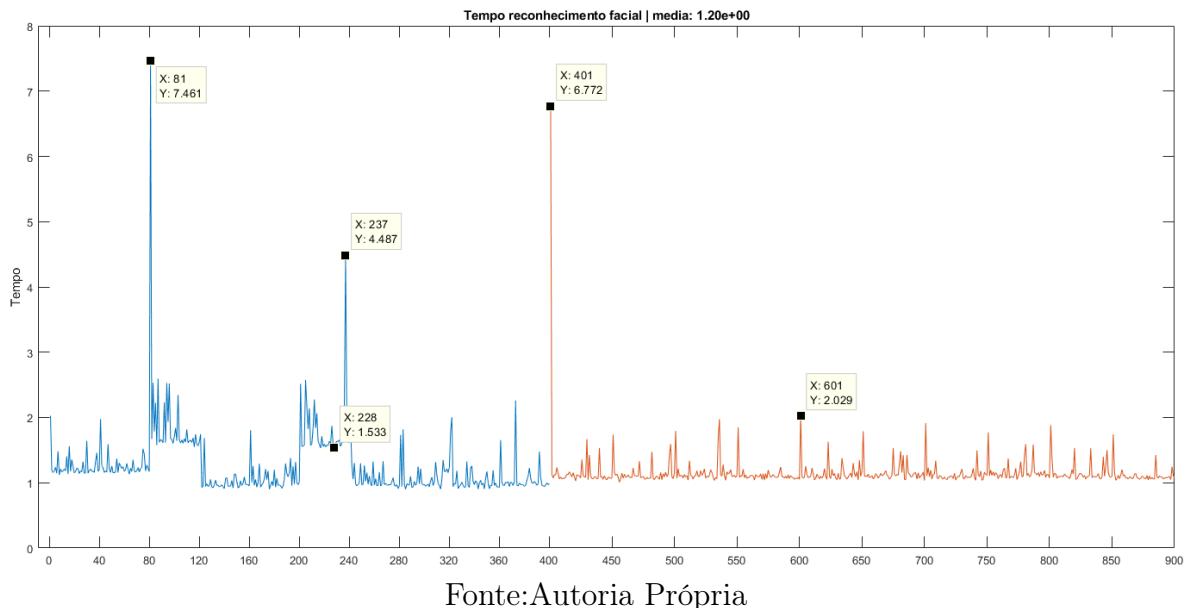
O objetivo desses dois processos foi realizar um número considerável de testes (processo 1) e verificar se existe alguma perda de performance em imagens obtidas em tempo real (processo 2).

As imagens de testes utilizadas tem as seguintes características.

- Usuário utilizado: Guilherme Cabral;
- Tamanho das imagens: 780x1040 pixels (imagem original) e 450x600 pixels (imagem tratada);
- Distância de aproximadamente 0,75m;
- Diferentes ângulos de análise.

O primeiro teste a ser realizado é o de variação do tamanho das imagens processadas, que foi realizado em duas etapas. O gráfico na Figura 68 seguir mostra os resultados obtidos para um total de 800 amostras.

Figura 68 – Teste da Variação do Tamanho da Imagem Enviada

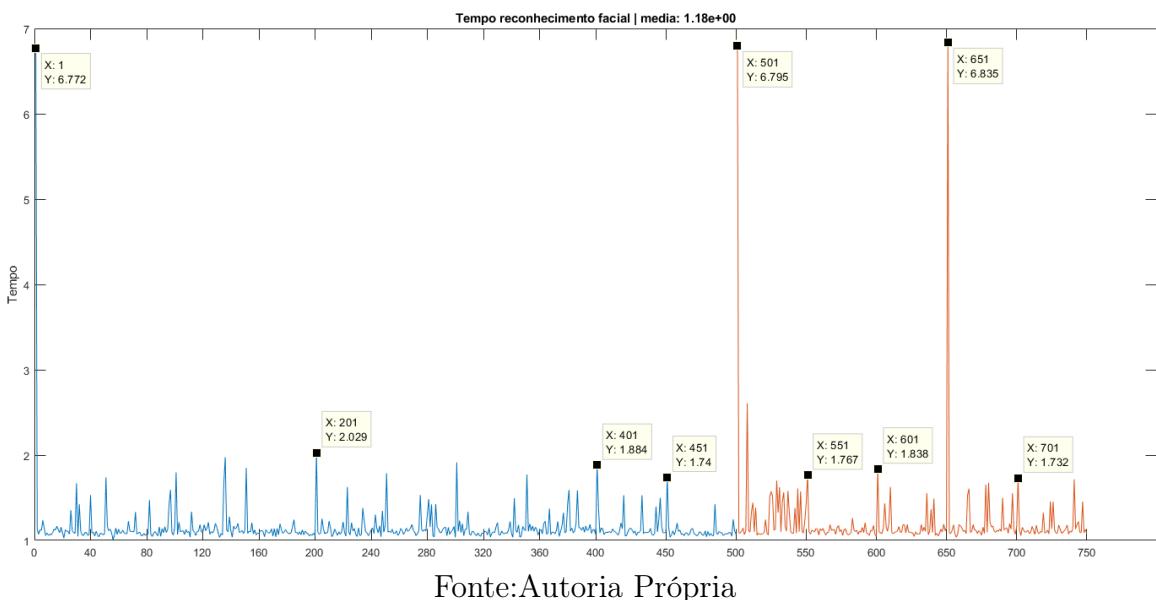


As fotos de tamanho original estão alocadas na primeira banda de amostras (0 - 400). Estas obtiveram uma instabilidade em relação ao tempo de resposta, uma vez que pode constatar a presença de picos, como na amostra 81 que teve como resultado um

pico de 7.641 segundos. Já na segunda banda (401 - 800) estão contidas as imagens que tiveram seus tamanhos tratados, que por sua vez apresentou resultados mais estáveis.

Apesar de existir uma divergência na amostra 401 essa não interfere nas conclusões do teste, pois esse pico ocorre com certa frequência na primeira imagem enviada a Cloud para o processamento e análise, após o início da conexão com a AWS, como pode ser observado no gráfico na Figura 69 seguir.

Figura 69 – Picos de tempo no início do teste



Para esse teste foram utilizadas 17 bandas, em que 9 delas possuem 40 amostras e as demais possuem 50 amostras. Assim, pode-se aferir através das amostras iniciais de cada banda esse pico no tempo em relação as demais realizadas no mesmo período. O motivo desse comportamento não está relacionado com a forma que é realizada a conexão. Na documentação do *Rekognition* também não é informado sobre essa peculiaridade.

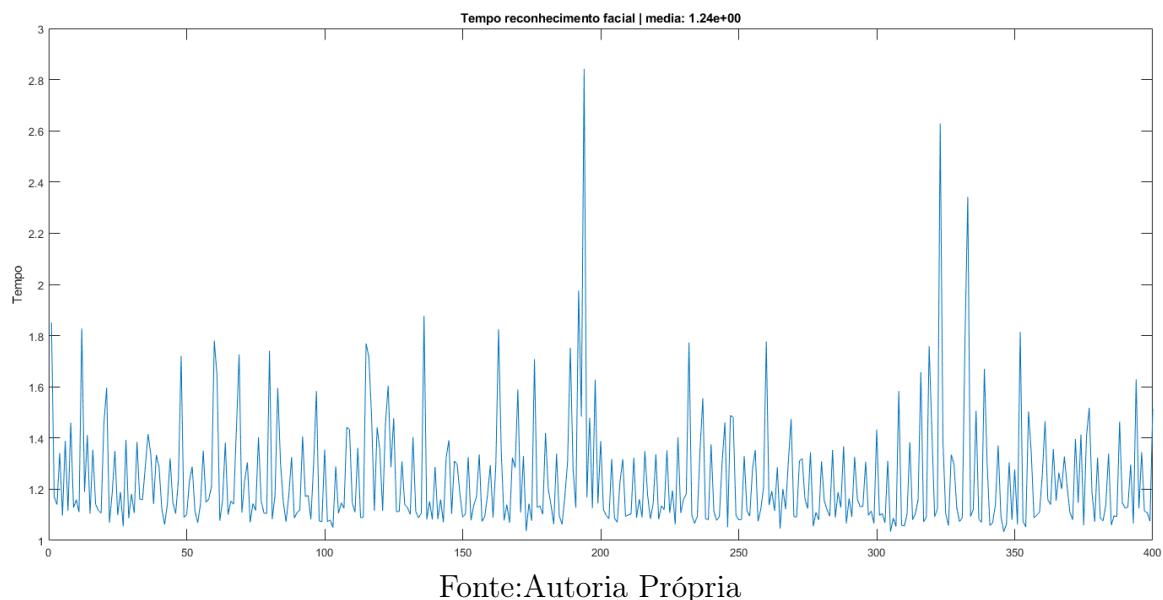
No entanto, esse problema não tem uma grande influência no funcionamento do MUSK, uma vez que o sistema só estaria propenso a ter tal comportamento na primeira imagem que envia para a AWS, após isso ele se estabiliza. Logo, esse pico pode ser facilmente corrigido via software em que, após a conexão com a AWS, é enviada uma imagem de teste para que o sistema se estabilize.

Em seguida foi realizado um teste para verificar se ocorria alterações no tempo devido a resultados de processamentos antigos. Até então foram realizados testes usando apenas uma persona o que poderia acarretar em um tempo de resposta cada vez menor. Para essa etapa foi efetuado o reconhecimento de diferentes pessoas famosas intercaladas entre si. As imagens delas foram tratadas usando o padrão desse projeto para que os resultados não ficassem enviesados.

- Usuário utilizado: Steve Jobs, Bill Gates, Jeff Bezos e Elon Musk;
- Tamanho das imagens: 450x600 pixels (imagem tratada);
- Cada rótulo utilizou 5 imagens para treino e determinação do vetor de característica.

Os resultados encontrados estão amostrados no gráfico da Figura 70 abaixo.

Figura 70 – Teste com variação da imagem de entrada

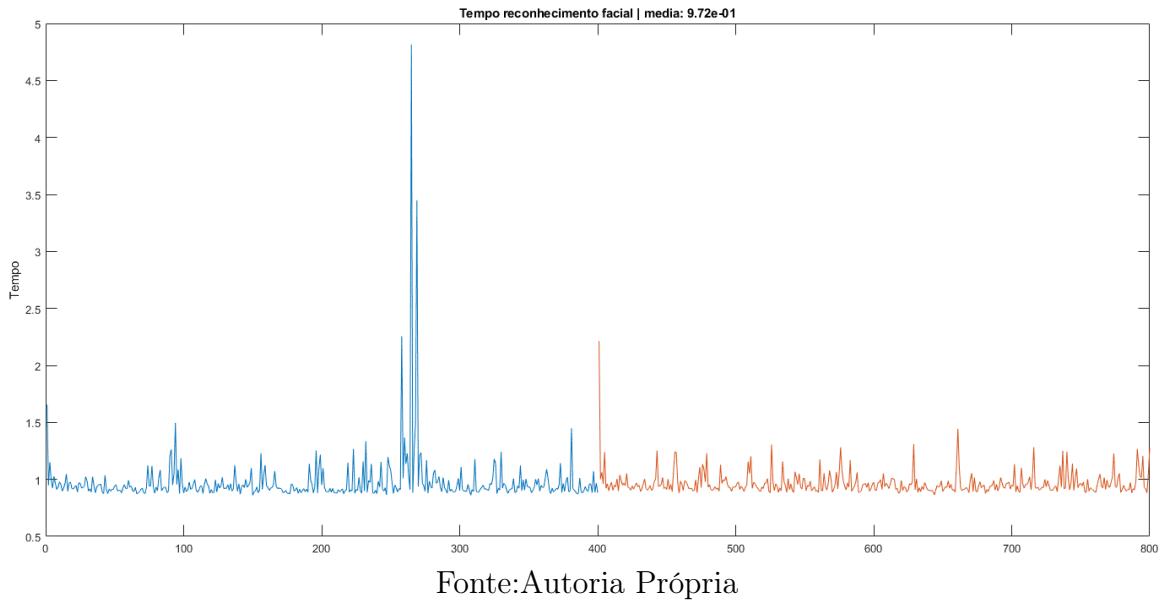


Fonte: Autoria Própria

Pode aferir-se que a API da AWS não é afetada pela variação de pessoas a serem reconhecidas, uma vez que o tempo médio de reconhecimento foi muito próximo dos realizados nos testes anteriores, sendo a diferença em milissegundos.

Por fim, foram realizados testes com fotos tiradas em tempo real utilizando o mesmo usuário Guilherme. O procedimento foi realizado tirando fotos frontais enquanto este estava realizando atividades no computador. Os resultados estão ilustrados na Figura 71 seguir.

Figura 71 – Teste em tempo real



Fonte: Autoria Própria

Assim, constatou-se a estabilidade da latência do processamento na *cloud*, além de um tempo médio de 0,92 segundos que é muito próximo dos obtidos com imagens pré-definidas. Observa-se que um conjunto de amostras obteve um tempo de resposta maior, com um máximo de 4,75 segundos, no entanto isso é resultado dos movimentos que o usuário realizou durante os testes, que afetou a qualidade do foco e a detecção do rosto, uma vez que o ângulo da face estava próximo do limiar permitido dificultando o processamento do *Rekognition*.

Por fim, em todos os testes executados analisou-se também dois aspectos importantes para um reconhecimento de faces eficiente. Os resultados obtidos foram especificados abaixo.

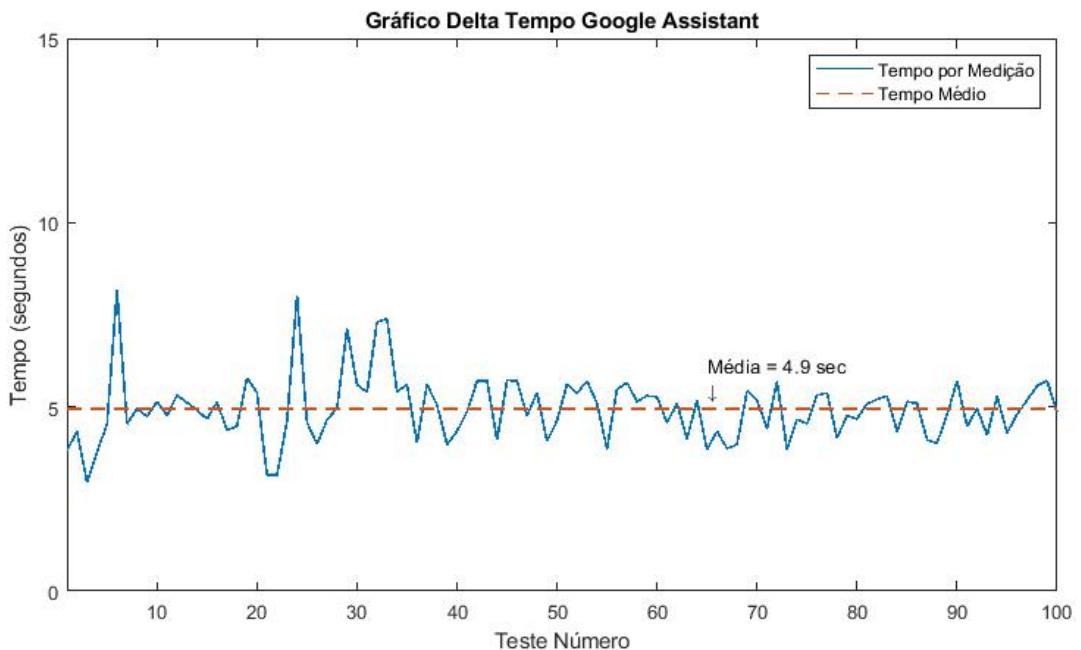
- Confiabilidade: valor em porcentagem de 0 a 100 que informa qual a confiança que se garante dos resultados retornados pela API. Em todos os testes realizados a confiabilidade foi de 100%.
- Similaridade: valor em porcentagem de 0 a 100 que informa qual a similaridade obtida entre a imagem enviada e o rótulo retornado (nome do usuário identificado). Em todos os testes foram obtidos uma similaridade de 100%.

5.3 Controle de Voz

Após a implementação do sistema de controle por voz buscou-se elaborar e realizar uma série de testes para averiguar o seu desempenho. No primeiro teste foi observado a latência do sistema em cem oportunidades, verificando-se assim um tempo máximo de

atendimento de 8,1772 segundos e um tempo mínimo de 2,9474 segundos. A média dos tempos obtidos foi de 4.92 segundos e observou-se que ao passar do tempo a variação dos atendimentos tendeu a diminuir como pode-se observar na Figura 72.

Figura 72 – Tempo de Requisição do Controle de Voz

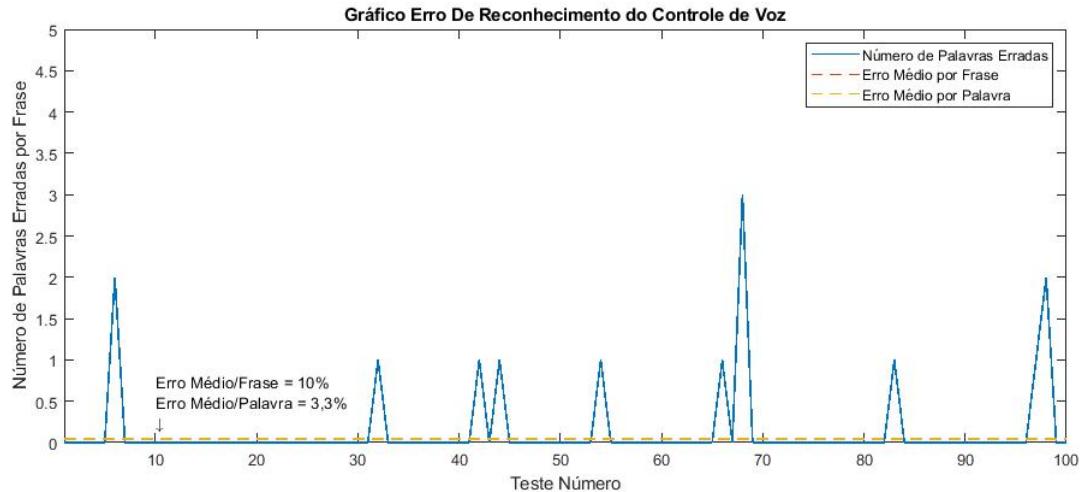


Fonte: Autoria Própria

No segundo teste decidiu-se observar a capacidade qualitativa do controle de entender corretamente as solicitações realizas. Para realizar este experimento realizou-se cem testes anotando a quantidade de palavras que foram compreendidas de forma erradas.

Aproximou-se que cada frase utilizada tinha em média três palavras e dentro de cada frase o erro máximo de palavras foi de três palavras, ou seja, de toda a sequência. Porém tal fato ocorreu apenas uma vez sendo que durante maior parte do tempo o erro foi nulo, ou seja, todas as palavras faladas foram compreendidas. O erro médio por frase foi de 0,1 (ou seja, 10%), enquanto o erro médio por palavra foi de 0,033 (ou seja, 3,3%) conforme apresentado no gráfico da Figura 73.

Figura 73 – Erro no Reconhecimento de Voz



Fonte: Autoria Própria

A partir da análise dos gráficos presente nas figuras 72 e 73 que o controle de voz possui uma elevada confiabilidade, baixa latência e um desempenho elevado. Porém como consequência disto este também necessita de uma quantidade elevada de processamento.

5.4 Performance do Sistema

Após a integração de todo o sistema, foi realizado um último teste para verificar qual o desempenho final do sistema durante seu funcionamento. Os objetivos desse teste são:

- Verificar se o sistema possui atrasos significativos nas respostas de suas funcionalidades, quando está realizando diversas tarefas ao mesmo tempo;
- Consultar o custo de memória e possíveis limitações da RPi em períodos de funcionamento extremo;
- Investigar possíveis aprimoramentos nas estruturas de software e hardware selecionadas e desenvolvidas;
- Verificar a existência de super aquecimento no assistente.

Com o intuito de analisar o funcionamento geral do sistema, foi proposto elevar o assistente a casos extremos de uso. A partir dessa necessidade, foi definido que o sistema realizaria diferentes ações ao mesmo tempo. Assim, diferentes funcionalidades foram testadas paralelamente, essas funcionalidades são:

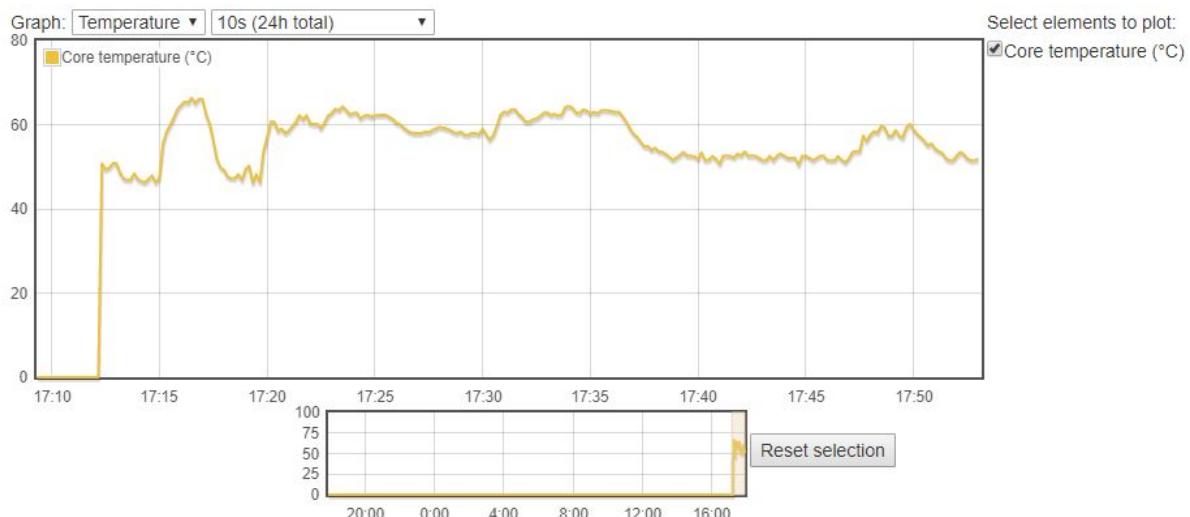
- Testes de comando de voz (pré-definidos e criados);

- Detecção e Reconhecimento Facial;
- Interação com a interface do usuário (LCD);
- Envio de requisições via interface Web para a movimentação dos motores, consulta da agenda e do sensor de temperatura e umidade.

Para realizar as medições de desempenho, foi utilizado o software RPi-Monitor que é específico para geração de métricas na raspberry. Uma vez que o objetivo desse capítulo é analisar os resultados obtidos, não será descrito o processo de instalação. No entanto o processo é simples e fácil de ser acessado na internet.

Os resultados obtidos nos testes de performance estão descritos e ilustrados na Figura 74 seguir. O período de avaliação do funcionamento do assistente foi de 38 minutos, com inicio às 17 horas e 15 minutos.

Figura 74 – Teste de temperatura



Fonte: Autoria Própria

O gráfico de variação de temperatura acima mostra que durante o período de execução do software do assistente pessoal, a RPi atingiu picos de quase 70C. Esses altos valores estão relacionados ao consumo do processador pelos programas que estão sendo executados. Uma vez que a temperatura normal de funcionamento é de 40 - 45C, observa-se que a raspberry atingiu alguns picos críticos de aquecimento.

Além de analisar a variação de temperatura outro fator importante a se observar é o consumo percentual da CPU. Para isso utilizou-se o mesmo programa apresentado anteriormente de maneira que obteve-se o gráfico presente na Figura 75.

Figura 75 – Teste de Consumo da CPU



Fonte: Autoria Própria

Através deste gráfico pode-se então perceber que em pequenos períodos de tempo houveram picos de 80%, tais períodos de tempo podem ser observados a partir dos dados obtidos a cada segundo. Pode-se perceber também que apesar dos picos a ocupação da CPU da Raspberry Pi mantém-se estável entre 40% a 50% aproximadamente. Tais valores por sua vez indicam um bom dimensionamento da distribuição do processamento da Raspberry, porém também indicam que a adição de outras funções sobre o assistente pessoal podem gerar tanto queda de desempenho na Raspberry quanto até mesmo a sua parada. Uma vez que elevados picos de processamento durante muito tempo podem gerar travamentos sobre a plataforma.

6 Conclusão

Este projeto foi concebido com o intuito de permitir o estudo de soluções em IoT para a construção de um assistente pessoal. O qual por sua vez deveria possibilitar a democratização das tecnologias utilizadas em assistentes pessoais comerciais.

Para isso, ao longo deste trabalho foi estudado diversas linguagens de programação e marcação, como Python, SQL, HTML, JavaScript e CSS, permitindo assim, a construção de um assistente pessoal que como qualquer outro sistema embarcado, integra elementos de software e hardware dentro de um complexo sistema.

Foi possível também validar a hipótese inicial da construção de um assistente pessoal de código aberto, flexível e acessível. A partir de bibliotecas gratuitas e públicas, tal como o Flask, OpenCV e Google Assistant. Sendo que estas contam com comunidades ativas.

Deve-se observar, no entanto, que o custo atribuído a execução do projeto foi maior do que o estimado. Gerando assim uma barreira na sua acessibilidade, a qual, porém, pode ser superada com a utilização de elementos de menor custo e consequentemente desempenho. Além disso, foi necessário integrar serviços pagos para a execução de tarefas, como identificação facial. O que demonstra, assim a dificuldade na construção de um assistente pessoal totalmente gratuito.

Quanto aos resultados obtidos, nota-se no capítulo anterior uma integração entre os serviços e programas sem interferências, apesar do elevado nível de processamento exigido em relação a Raspberry Pi 3, que atingiu picos de 80%, junto com picos de aquecimento da placa de 60°C. Essa limitação quanto ao hardware só pode ser contornada com o avanço de tecnologias de processamento de baixo custo.

A utilização de periféricos USB permite uma flexibilização na qualidade em relação ao custo desempenho, uma vez que, isso permite a utilização de dispositivos de menor custo ou de maior qualidade, caso necessário.

Por fim, pode se afirmar o sucesso deste projeto, uma vez que, este apresentou uma integração dos serviços de forma a não gerar problemas de desempenho entre eles, junto com um consumo de recursos do sistema sem interferências. Tal como, comando de voz, reconhecimento facial, interface Web e visual, além da utilização de dispositivos externos (sensores, atuadores e display LCD). Sendo que estes apresentaram um bom desempenho devido a latência nos tempos de resposta dos serviços, e requisições do sistema, e confiabilidade nos resultados obtidos.

6.1 Aprimoramentos Futuros

Devido ao seu objetivo principal que é de servir como base para desenvolvimentos futuros na área, o assistente pessoal, ainda tem muito a evoluir. Entre alguns dos elementos que elevariam o potencial deste assistente pessoal estão: A integração do assistente pessoal a um servidor, permitindo assim o acesso de diferentes ambientes. Sendo que neste caso deverá haver um aumento de preocupação com o nível de segurança oferecido pelo site, para que os seus usuários tenham os seus dados protegidos.

A criação de um aplicativo *Mobile* para facilitar a utilização das atuais ferramentas. Sendo que esta é uma grande oportunidade para se utilizar novas ferramentas e tecnologias tal como o React Native, Flutter e Xamarin.

Por último recomenda-se um estudo de otimização do software para permitir tanto a expansão das suas ferramentas quanto para facilitar o acesso fazendo com que o mesmo possa ser compilado em hardwares de menor potência.

Referências

- [1] Unlocking the potential of the internet of things. <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world>. Accessed: 2019-10-15. Citado na página 13.
- [2] Smart home technology poised for blockbuster growth. <https://www.statista.com/chart/15736/smart-home-market-forecast/>. Accessed: 2019-10-15. Citado na página 13.
- [3] Apple future computer knowledge navigator. <https://www.businessinsider.com/apple-future-computer-knowledge-navigator>. Accessed: 2019-10-15. Citado na página 13.
- [4] Assistentes pessoais virtuais ganham o mundo corporativo. <https://cio.com.br/assistentes-pessoais-virtuais-ganham-o-mundo-corporativo/>. Accessed: 2019-10-15. Citado na página 13.
- [5] Assistentes pessoais digitais. <https://oglobo.globo.com/economia/assistentes-pessoais-digitais-1-19383632>. Accessed: 2019-10-15. Citado na página 13.
- [6] Internet das coisas: Um desenho do futuro. <https://www.proof.com.br/blog/internet-das-coisas/>. Accessed: 2019-10-15. Citado na página 13.
- [7] Internet of things global standards initiative. <https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>. Accessed: 2019-10-15. Citado na página 13.
- [8] W STALLINGS. *Arquitetura e organização de computadores*. Pearson Prentice-Hall, 10^a ed., 2017. Citado na página 16.
- [9] Internet of things global standards initiative. <https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>. Accessed: 2019-10-15. Citado na página 16.
- [10] 5 áreas mais impactadas pela internet das coisas. <https://cio.com.br/5-areas-mais-impactadas-pela-internet-das-coisas/>. Accessed: 2019-10-15. Citado na página 17.
- [11] Daniel Thomazini. *Sensores industriais: fundamentos e aplicações*. Saraiva Educação SA, 2018. Citado na página 17.

- [12] Sinal analógico x sinal digital. <https://www.embarcados.com.br/sinal-analogico-x-sinal-digital/>. Accessed: 2019-10-14. Citado na página 17.
- [13] Sensores e atuadores. <https://www.mecanicaindustrial.com.br/374-para-que-servem-os-atuadores/>. Accessed: 2019-10-15, Author: Prof. Maurílio J. Inácio. Citado na página 17.
- [14] O que é a computação em nuvem? <https://aws.amazon.com/pt/what-is-cloud-computing/>. Accessed: 2019-10-15, Author: Prof. Maurílio J. Inácio. Citado na página 18.
- [15] O que é open source? <https://canaltech.com.br/produtos/0-que-e-open-source/>. Accessed: 2019-10-15. Citado na página 18.
- [16] Maik Schmidt. *Raspberry Pi: A Quick-Start Guide*. Pragmatic Bookshelf, 2014. Citado 2 vezes nas páginas 20 e 31.
- [17] Saiba tudo sobre o raspberry pi 3 e o que ele representa para o mercado. <https://canaltech.com.br/hardware/saiba-tudo-sobre-o-raspberry-pi-3-59065/>. Accessed: 2019-10-14. Citado na página 21.
- [18] Processador arm. http://www.dca.fee.unicamp.br/~lboccato/topicos_3.1_processador_ARM.pdf. Accessed: 2018-06-20. Citado na página 21.
- [19] Display lcd tft touch 3.5inc raspberry pi. <https://www.filipeflop.com/produto/display-lcd-tft-touch-3-5-raspberry-pi/>. Accessed: 2019-10-14. Citado na página 21.
- [20] 3.5inch rpi display. http://www.lcdwiki.com/3.5inch_RPi_Display. Accessed: 2019-10-14. Citado na página 21.
- [21] PIR HC-SR501. Motion detector datasheet. *Product Description*. Citado na página 23.
- [22] Warren W Gay. Dht11 sensor. In *Experimenting with Raspberry Pi*, pages 1–13. Springer, 2014. Citado na página 25.
- [23] Manual webcam goldship 3817. Manufacturer: GoldShip, Inc, LeaderSheep. Citado na página 26.
- [24] Microfone universal mini usb 2.0. Manufacturer: Mini, Inc. Citado na página 27.
- [25] O que é servomotor? controlando um servo com arduino. <https://portaldasilicio.com.br/o-que-e-servomotor/>. Accessed: 2019-10-14. Citado na página 29.

- [26] Andrew S Tanenbaum and Herbert Bos. *Modern operating systems*. Pearson, 2015. Citado 2 vezes nas páginas 30 e 31.
- [27] About raspbian. <https://www.raspbian.org/RaspbianAbout>. Accessed: 2019-10-14. Citado na página 31.
- [28] How to build your own digital assistant with a raspberry pi. <https://www.androidauthority.com/build-google-assistant-raspberry-pi-770296/>. Accessed: 2019-10-14. Citado 2 vezes nas páginas 31 e 76.
- [29] Introdução a Classes e Métodos em Python kernel description. <http://pythonclub.com.br/introducao-classes-metodos-python-basico.html>. Accessed: 2018-06-22. Citado na página 32.
- [30] Como funcionam as aplicações web. <https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888>. Accessed: 2019-10-14. Citado na página 32.
- [31] Métodos de requisição http. <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Accessed: 2019-10-14. Citado na página 32.
- [32] What does “micro” mean? <https://flask-doc.readthedocs.io/en/latest/foreword.html>. Accessed: 2019-10-14. Citado na página 33.
- [33] Python flask introdução. <http://devfuria.com.br/python/flask/>. Accessed: 2019-10-14. Citado na página 33.
- [34] Qual a diferença entre página web, site, servidor web e mecanismo de busca? https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/Pages_sites_servers_and_search_engines. Accessed: 2019-06-20. Citado na página 33.
- [35] Definition of: dynamic web page. <https://web.archive.org/web/20170117040526/https://www.pcmag.com/encyclopedia/term/42199/dynamic-web-page>. Accessed: 2019-06-20. Citado na página 33.
- [36] Html - standards. <https://www.w3.org/standards/>. Accessed: 2019-06-20. Citado na página 34.
- [37] Addison WESLEY. *A history of HTML*. 1998. Citado na página 34.
- [38] O que é css? guia básico para iniciantes. <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>. Accessed: 2019-06-20. Citado na página 34.

- [39] Henry F KORTH and A SILBERCHATZ. Sistemas de banco de dados. rev. Citado na página 36.
- [40] Christopher J Date. *Introdução a sistemas de bancos de dados*. Elsevier Brasil, 2004. Citado na página 36.
- [41] Jane Laudon, Kenneth; Laudon. *Sistemas de Informação Gerenciais*. Pearson Brasil, 2011. Citado na página 37.
- [42] Dados não relacionais e nosql. <https://docs.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data>. Accessed: 2019-10-14. Citado na página 37.
- [43] Sql. <https://www.sqlite.org/features.html>. Accessed: 2018-05-06. Citado na página 38.
- [44] O que é visão computacional? <http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>. Accessed: 2019-10-14. Citado na página 38.
- [45] O que é visão computacional? <https://blogbrasil.comstor.com/o-que-e-visao-computacional>. Accessed: 2019-10-14. Citado na página 38.
- [46] Como funciona o reconhecimento facial. <https://www.techtudo.com.br/artigos/noticia/2012/04/como-funciona-o-reconhecimento-facial.html>. Accessed: 2019-10-14. Citado na página 39.
- [47] Opencv about. <https://opencv.org/about/>. Accessed: 2019-10-14. Citado na página 39.
- [48] Opencv documentation. <https://docs.opencv.org/4.1.1/d1/dfb/intro.html>. Accessed: 2019-10-14. Citado na página 39.
- [49] O que é a computação em nuvem? <https://aws.amazon.com/pt/what-is-cloud-computing/>. Accessed: 2019-10-14. Citado na página 40.
- [50] Amazon rekognition. <https://aws.amazon.com/pt/rekognition/>. Accessed: 2019-10-14. Citado na página 40.
- [51] Como funciona o reconhecimento de voz? <https://www.tecmundo.com.br/curiosidade/3144-como-funciona-o-reconhecimento-de-voz-.htm>. Accessed: 2019-10-14. Citado na página 41.
- [52] Actions on google: Desenvolvendo actions para o google assistant do zero. <https://medium.com/@wmessiascavalcanti/>

- [actions-on-google-desenvolvendo-actions-para-o-google-assistant-do-zero-75a7ae](https://www.google.com/search?q=actions-on-google-desenvolvendo-actions-para-o-google-assistant-do-zero-75a7ae)
Accessed: 2019-10-14. Citado na página 42.
- [53] Change your Network Card MAC Address on Ubuntu kernel description. <https://www.howtogeek.com/howto/ubuntu/change-your-network-card-mac-address-on-ubuntu/>. Accessed: 2018-06-22. Citado na página 45.
- [54] Dynamic Host Configuration Protocol (DHCP) kernel description. <https://www.raspberrypi.org/learning/networking-lessons/lesson-3/plan/>. Accessed: 2018-06-22. Citado na página 45.
- [55] MAC address spoofing kernel description. https://wiki.archlinux.org/index.php/MAC_address_spoofing. Accessed: 2018-06-22. Citado na página 45.
- [56] SSH (SECURE SHELL) kernel description. <https://www.raspberrypi.org/documentation/remote-access/ssh/>. Accessed: 2018-06-22. Citado na página 46.
- [57] Como montar um servidor web com o Raspberry Pi kernel description. <http://www.raspberrypiportugal.pt/montar-um-servidor-web-raspberry-pi/>. Accessed: 2018-06-22. Citado na página 47.
- [58] Sites responsivos com muita rapidez. https://www.adobe.com/br/products/dreamweaver.html?gclid=Cj0KCQjw3JXtBRC8ARIsAEBHg4k4GxQiSH1z0I7fXA_YecH2e3Up9_ExjntHECwgIrRZDE1NihMCC0YaAuUZEALw_wcB&sdid=KQPQE&mv=search&ef_id=Cj0KCQjw3JXtBRC8ARIsAEBHg4k4GxQiSH1z0I7fXA_YecH2e3Up9_ExjntHECwgIrRZDE1NihMCC0YaAuUZEALw_wcB:G:s&s_kwcid=AL!3085!3!301784449881!e!!g!!dreamweaver. Accessed: 2019-10-14. Citado na página 48.
- [59] Raspberry Temp. Server kernel description. <http://raspberrywebserver.com/cgiscripting/rpi-temperature-logger/building-a-web-user-interface-for-the-temperature-monitor.html>. Accessed: 2018-06-22. Citado na página 50.
- [60] Raspberry Forms Server kernel description. <http://raspberrywebserver.com/cgiscripting/web-forms-with-python.html>. Accessed: 2018-06-22. Citado na página 50.
- [61] XMLHttpRequest kernel description. https://www.w3schools.com/xml/xml_http.asp. Accessed: 2018-06-22. Citado na página 50.
- [62] Enviando e recebendo emails com python. <https://humberto.io/pt-br/blog/enviando-e-recebendo-emails-com-python/>. Accessed: 2019-10-14. Citado na página 64.

- [63] Definição de preço do amazon rekognition. <https://aws.amazon.com/pt/rekognition/pricing/?nc=sn&loc=4>. Accessed: 2019-10-14. Citado na página 73.
- [64] Configurar uma conta da aws e criar um usuário do iam. https://docs.aws.amazon.com/pt_br/rekognition/latest/dg/setting-up.html. Accessed: 2019-10-14. Citado na página 74.
- [65] Aws sdk para python (boto3). <https://aws.amazon.com/pt/sdk-for-python/>. Accessed: 2019-10-14. Citado na página 74.
- [66] Recomendações para imagens de entrada de reconhecimento facial. https://docs.aws.amazon.com/pt_br/rekognition/latest/dg/recommendations-facial-input-images.html. Accessed: 2019-10-14. Citado na página 74.
- [67] Available services. <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html>. Accessed: 2019-10-14. Citado na página 75.

Apêndices

APÊNDICE A – Gerenciador do Servidor: *server.py*

```
1  from flask import Flask, render_template, Response, request, url_for, abort
2      , redirect
3  from flask_sqlalchemy import SQLAlchemy
4  from datetime import datetime
5  import sqlite3
6  import os
7  import threading
8  from sensor import sensor
9  import camera
10 import main
11 import servoMotor

13 tccCamera = camera.camera()
14 tccMotor = servoMotor.servoMotor()
15 tccMotor.runConfiguration()

17 def thread_cam (data):
18     print("entrou na thread")
19     main.run(tccCamera)

21 t = threading.Thread(target=thread_cam , args=( "null" ,))
22 t.start()

23 def thread_voice (data):
24     print("entrou na thread_voice")
25     os.system("sudo bash try_bash_.sh")

27 t = threading.Thread(target=thread_voice , args=( "null" ,))
28 t.start()

31 def thread_kivy(data):
32     print("entrou na thread kivy")
33     os.system("sudo python kivy_gui.py")

35 t = threading.Thread(target=thread_kivy , args=( "null" ,))
36 t.start()

37
```

```
39 def generateVideo(cam):
40     while True:
41         frame = cam.sendVideo()
42         yield (b'--frame\r\n'
43                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
44
45
46 # Starting Flask Server and Sqlite3
47 app = Flask(__name__)
48 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///home/pi/Desktop/
49                                         tcc_files/opencv_aws/todo.db'
50 db = SQLAlchemy(app)
51
52
53 class Todo(db.Model):
54     id = db.Column(db.Integer, primary_key=True)
55     title = db.Column(db.String(200))
56     start = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
57     complete = db.Column(db.Boolean)
58
59
60 @app.route("/")
61 def index():
62     return render_template('home_entry.html')
63
64 @app.route('/inside')
65 def index2():
66     return render_template('home_inside.html')
67
68 @app.route('/contact_pag')
69 def contact():
70     return redirect(url_for('index'))
71
72 @app.route('/camera')
73 def camera():
74     return render_template('camera.html')
75
76 @app.route('/cameraview')
77 def cameraview():
78     return Response(generateVideo(tccCamera), mimetype='multipart/x-mixed-
79     replace; boundary=frame')
80
81 @app.route('/panleft')
82 def move2():
83     tccMotor.servo_position(0, -1)
84     return redirect(url_for('camera.html'))
```

```
85 @app.route('/panright')
86 def move3():
87     tccMotor.servo_position(0,1)
88     return redirect(url_for('camera.html'))
89
90 @app.route('/temperatura')
91 def temp():
92     umidade, temperatura = sensor.readDht11();
93     templateData = {'umid': "10", 'temp': "10"}
94     return render_template('temperaturahtml.html', **templateData)
95
96 @app.route('/agenda_pag')
97 def agenda():
98     todos = Todo.query.all()
99     incomplete = Todo.query.filter_by(complete=False).all()
100    complete = Todo.query.filter_by(complete=True).all()
101    return render_template('Agenda.html', incomplete=incomplete, complete=complete)
102
103 @app.route('/add', methods=['POST'])
104 def add():
105     todo = Todo(title=request.form['todoitem'], start=datetime.strptime(
106         request.form['date'], '%Y-%m-%d'), complete=False)
107     db.session.add(todo)
108     db.session.commit()
109     return redirect(url_for('agenda'))
110
111 @app.route('/complete/<id>')
112 def complete(id):
113     print("id obtido: " + str(id))
114     todo = Todo.query.filter_by(id=int(id)).first()
115     todo.complete = True
116     db.session.commit()
117
118 if __name__ == "__main__":
119     app.run(host='0.0.0.0', port=3000, debug=False)
```

APÊNDICE B – Programa *main.py*

```
2 import numpy as np
3 import cv2
4 import time
5 import json
6 import personaVerification
7 import awsConnection
8 import sensor
9 import security
10 import camera
11
12 def run(tccCamhttps://www.overleaf.com/download/project/5
13         dbaeae75ec01100010104c4/build/16e3931c71c-df1cd0a65045b15c/output/output
14         .pdf?compileGroup=standard&clsiserverid=clsi-pre-emp-pq3c&cache_bust
15         =1572917593506&popupDownload=true):
16     # Creating objects
17     tccAws = awsConnection.awsConnection()
18     tccPersona = personaVerification.personaVerification()
19     tccSensor = sensor.sensor()
20     tccSecurity = security.security()
21     print("antes do cv2")
22     tccCamera = tccCam
23     print("depois do cv2")
24
25     # Connecting to AWS Rekognition
26     tccAws.getRekognitionClient()
27
28     # Motion callback function
29     def motionCallback(channel):
30         print("[+] Motion detected")
31         return tccSecurity.sendEmail()
32
33     # Starting sensors
34     tccSensor.runConfiguration(motionCallback)
35
36     while(True):
37         frame = tccCamera.runFaceDetector()
38         if(tccCamera.flag):
39             tccPersona.faceMatch(tccAws.awsRekognitionClient)
```

```
40     cv2.imshow( 'frame' , frame )
41
42     #stop when 'q' is pressed
43     if cv2.waitKey(20) & 0xFF == ord( 'q' ):
44         break
45
46     #release the capture on the end
47     cap.release()
48     cv2.destroyAllWindows()
49     tccSensor.cleanSensorPorts()
```

APÊNDICE C – Programa *awsConnection.py*

```
1 import boto3 as b3
3
5 class awsConnection():
6     def __init__(self):
7         self.awsRekognitionClient = 0
8         self.key_id = "key_id"
9         self.secret_key = "secret_key"
10        self.aws_region = "region"
11
12    def getRekognitionClient(self):
13        self.awsRekognitionClient = b3.client('rekognition', aws_access_key_id
14            = self.key_id, aws_secret_access_key = self.secret_key, region_name
15            = self.aws_region)
16        print("[+] Connected to AWS Rekognition")
```

APÊNDICE D – Programa *camera.py*

```
1 import numpy as np
3 import cv2
5
5 class camera():
7     def __init__(self):
8         self.lineRect = 2
9         self.colorRect = (0,255,0)
10        self.captureCamera = cv2.VideoCapture(0)
11        self.faceCascade = cv2.CascadeClassifier('cascade/data/
12            haarcascade_frontalface_default.xml')
13        self.faces = 0
14        self.gray_frame = 0
15        self.frame = 0
16        self.flag = False
17
17    def getFrame(self):
18        #capture frame
19        return self.captureCamera.read()
20
21    def destroy(self):
22        return self.captureCamera.release()
23
24    def getImage(self):
25        for (x,y,w,h) in self.faces:
26            print(x,y,w,h)
27            #Cords
28            end_cord_x = x+w+50; end_cord_y = y+h+50;
29            ini_cord_x = x-25; ini_cord_y = y-25;
30            #roi – region of interest
31            roi_gray = self.gray_frame[ini_cord_y:end_cord_y, ini_cord_x:
32                end_cord_x]
33            roi_color = self.frame[ini_cord_y:end_cord_y, ini_cord_x:end_cord_x]
34            roi_gray_res = cv2.resize(roi_gray, dsize=(420, 420), interpolation=
35                cv2.INTER_CUBIC)
36            cv2.imwrite("img1.jpg", roi_gray_res)
37            cv2.rectangle(self.frame,(x,y), (end_cord_x,end_cord_y), self.
38                colorRect, self.lineRect)
39            self.flag = True
```

```
39  def runFaceDetector(self):  
40      self.flag = False  
41      ret, self.frame = self.getFrame();  
42      #turn into gray  
43      self.gray_frame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)  
44      self.faces = self.faceCascade.detectMultiScale(self.gray_frame,  
45          scaleFactor=1.5, minNeighbors=8)  
46      self.getImage();  
47      return self.frame  
48  
48  def sendVideo(self):  
49      success, image = self.getFrame()  
50      ret, jpg = cv2.imencode('.jpg', image)  
51      return jpg.tobytes()
```

APÊNDICE E – Programa *kivy_gui.py*

```
1  # -*- coding: utf-8 -*-
3
4  import kivy
5  from kivy.app import App
6  from kivy.lang import Builder
7  from kivy.uix.screenmanager import ScreenManager, Screen, NoTransition
8  from kivy.uix.button import Button
9  from kivy.clock import Clock
10 from kivy.uix.widget import Widget
11 from kivy.uix.floatlayout import FloatLayout
12 from kivy.properties import ObjectProperty
13 from kivy.uix.boxlayout import BoxLayout
14 from kivy.uix.scrollview import ScrollView
15
16 from datetime import datetime
17 import databaseSql
18 import time
19 import os
20 import commands
21 import sensor
22
23 # ----- CONFIG KIVY -----
24 kivy.require('1.8.0')
25
26 # You can create your kv code in the Python file
27 Builder.load_string("""
28
29 <ScreenOne@Button>:
30
31     background_color: 0,0,0,1
32
33     on_press:
34         root.manager.current = 'screen_two'
35
36     Image:
37         source: 'skull1.jpg'
38         y: self.parent.y + self.parent.height - 600
39         x: self.parent.x + 0
40         size: 35,35
41         allow_stretch: True
```

```
Label:  
43     text: "M.U.S.K"  
44     font_size: '40sp'  
45     bold: True  
46  
47     <ScreenTwo@Button>:  
48  
49         background_color: 0,0,0,1  
50  
51         on_press:  
52             root.manager.current = 'screen_three'  
53  
54         <Button:  
55             background_color: 0,0,0,1  
56             text: root.init_var()  
57             size: 150,50  
58             font_size: 36  
59             pos_hint: {"center_x": .5, "center_y": .65}  
60  
61     <ScreenThree@Button>:  
62  
63         background_color: 0,0,0,1  
64  
65         on_press:  
66             root.manager.current = 'screen_one'  
67  
68         <Button:  
69             background_color: 0,0,0,1  
70             text: root.registrador_tarefas  
71             size: 150,50  
72             font_size: 36  
73  
74     >>>  
75  
76  
77  
78  
79 tccDatabase = databaseSql.databaseSql()  
80 tccSensor = sensor.sensor()  
81  
82 class ScreenOne(Screen, Button):  
83  
84     def __init__(self, **kwargs):  
85         super(ScreenOne, self).__init__(**kwargs)  
86  
87     def callback(self, dt):  
88         calling this funcition
```

```
89         screen_manager.transition = NoTransition()
90         screen_manager.current = 'screen_two'
91
93 class ScreenTwo(Screen, Button):
95
95     def __init__(self, **kwargs):
96         super(ScreenTwo, self).__init__(**kwargs)
97         Clock.schedule_interval(self.callback, 29)
98
99     def init_var(self):
100
101         umid, temp = tccSensor.readDht11();
102         umid = "Umidade: " + str(umid) + "% \n"
103         temp = "Temperatura: " + str(temp) + " C \n"
104
105         now = datetime.now()
106         date = "\n" + "\n" + "\n" + "Hoje: " + str(now.day) + "/" + str(
107             now.month) + "/" + str(now.year) + "\n"
108         time = "Horas: " + str(now.hour) + ":" + str(now.minute) + ":" +
109             str(now.second)
110
111         registrador_umid_e_temp = ObjectProperty(None)
112         registrador_umid_e_temp = temp + umid + date + time
113         return registrador_umid_e_temp
114
115     def callback(self, dt):
116         # print('In Callback') # Test - The timer is actually calling this
117             # function
118
119         screen_manager.transition = NoTransition()
120         screen_manager.current = 'screen_one'
121
122     class ScreenThree(Screen, Button):
123
124         title = tccDatabase.read_string_db()
125         len_title = len(title)
126
127         tarefas = '';
128         for i in range(len_title):
129             tarefas = tarefas + str(title[i]) + '\n',
130
131         registrador_tarefas = ObjectProperty(None)
132         registrador_tarefas = tarefas
```

```
133
135
137     def __init__(self, **kwargs):
138
139         super(ScreenThree, self).__init__(**kwargs)
140         layout = BoxLayout(orientation="vertical", size_hint_y=None)
141
142         btn_volta = Button(text="Volta",
143                             pos_hint = {"left": 1, "bottom": 1},
144                             size=(810, 700),
145                             size_hint=(None, None),
146                             background_color=(0, 0, 0, 1),
147                             font_size=(36),
148                             on_press=self.Press_auth)
149         layout.add_widget(btn_volta)
150
151         root = ScrollView()
152         root.add_widget(layout)
153         self.add_widget(root)
154
155     def Press_auth(self, instance):
156         screen_manager.transition = NoTransition()
157         screen_manager.current = 'screen_one'
158         print(str(instance))
159
160     # The ScreenManager controls moving between screens
161     screen_manager = ScreenManager()
162
163     # Add the screens to the manager and then supply a name
164     # that is used to switch screens
165     screen_manager.add_widget(ScreenOne(name="screen_one"))
166     screen_manager.add_widget(ScreenTwo(name="screen_two"))
167     screen_manager.add_widget(ScreenThree(name="screen_three"))
168
169     class KivyTut2App(App):
170
171         def build(self):
172             return screen_manager
173
174     if __name__ == '__main__':
175         KivyTut2App().run()
```

APÊNDICE F – Programa

personaVerification.py

```
2 import boto3 as b3

4 class personaVerification():
5     def __init__(self):
6         self.name = 0
7         self.similarity = 0
8         self.confidence = 0
9         self.picture = 'img1.jpg'
10        self.collectionName = 'collectionTcc'

12    def checkFace(self, client):
13        face_detected = False
14        with open(self.picture, 'rb') as image:
15            response = client.detect_faces(Image={'Bytes': image.read()})
16            if(not response['FaceDetails']):
17                face_detected = False
18            else:
19                face_detected = True
20        return face_detected, response

22    def checkMatches(self, client):
23        face_matches = False
24        with open(self.picture, 'rb') as image:
25            response = client.search_faces_by_image(CollectionId=self.
26                collectionName, Image={'Bytes': image.read()}, MaxFaces=1,
27                FaceMatchThreshold=95)
28            if (not response['FaceMatches']):
29                face_matches = False
30            else:
31                face_matches = True
32        return face_matches, response

33    def faceMatch(self, client):
34        result, resp = self.checkFace(client);
35        if(result):
36            resu, res = self.checkMatches(client);
37            if resu:
38                self.name = res['FaceMatches'][0]['Face'][ 'ExternalImageId']
```

```
38     self.similarity = res[ 'FaceMatches' ][0][ 'Similarity' ]
  39     self.confidence = res[ 'FaceMatches' ][0][ 'Face' ][ 'Confidence' ]
40 else :
41     self.name = 'unknown'
42     self.similarity = 0
43     self.confidence = 0
44     print ("[+] Detected Person: " + str(self.name))
45     print ("[+] Similarity: " + str(self.similarity))
46     print ("[+] Confidence: " + str(self.confidence))
47     print (" ")
48 else :
49     print ("[+]No faces detected")
```

APÊNDICE G – Programa *security.py*

```
2 import smtplib
4
4 class security():
5     def __init__(self):
6         self.receiver = 'gui.cabral201@gmail.com'
7         self.topic = 'MUSK Security Service'
8         self.text = 'Motion Detected'
9         self.sender = 'projetosd373@gmail.com'
10        self.password = 'projetoel373'
11        self.message = '\r\n'.join(['From: %s' % self.sender, 'To: %s' % self.
12                                   receiver, 'Subject: %s' % self.topic, '', '%s' % self.text])
13        self.server = smtplib.SMTP()
14
14    def sendEmail(self):
15        self.server.connect('smtp.gmail.com', '587')
16        self.server.starttls()
17        self.server.login(self.sender, self.password)
18        self.server.sendmail(self.sender, self.receiver, self.message)
19        self.server.quit()
20        return print("[+]The email was send")
21
22    def sendAlert(self):
23        print("[+]Send an alert message to user's phone")
```

APÊNDICE H – Programa *sensor.py*

```
1 import RPi.GPIO as GPIO
3 import Adafruit_DHT
5
7 #def testeCall(channel):
# print("deu bom")
9
9 class sensor():
11     def __init__(self):
12         self.pinoMotion = 24
13         self.pinoTemperature = 26
14         self.sensorTemperatureType = Adafruit_DHT.DHT11
16
15     def runConfiguration(self, motionFunction):
16         GPIO.setmode(GPIO.BOARD)
17         GPIO.setup(self.pinoMotion, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
18         GPIO.add_event_detect(self.pinoMotion, GPIO.RISING, callback=
19             motionFunction, bouncetime=300)
20         print("[+] Sensors configured")
21
21     def readDht11(self):
22         umidade, temperatura = Adafruit_DHT.read_retry(self.
23             sensorTemperatureType, self.pinoTemperature)
24         return umidade, temperatura
25         #return "10", "10"
26
27     def cleanSensorPorts(self):
28         GPIO.cleanup()
```

APÊNDICE I – Programa *servoMotor.py*

```
2 import Adafruit_PCA9685
4
5 class servoMotor():
6
7     def __init__(self):
8         self.pwm = Adafruit_PCA9685.PCA9685()
9         self.servo_min = 150
10        self.servo_max = 600
11        self.memory = [0,0]
12
13    def runConfiguration(self):
14        self.pwm.set_pwm_freq(60)
15
16    def step(x):
17        y = 375 + 225*x
18        if y>=600:
19            y = 600
20        if y<=150:
21            y = 150
22        return y
23
24    def servo_position(self, motor, pos):
25
26        if pos == 1:
27            self.memory[0] += 1
28            print("pos = 1")
29        if pos == -1:
30            self.memory[0] -= 1
31
32        pulse = self.step(self.memory[0])
33        self.pwm.set_pwm(0, 0, pulse)
```

APÊNDICE J – Código Página Principal

HTML

```
1<!doctype html>
2<html lang="pt-BR">
3<head>
4<meta charset="utf-8">
5<meta http-equiv="X-UA-Compatible" content="IE=edge">
6<meta name="viewport" content="width=device-width, initial-scale=1">
7<title>MUSK</title>
8<link href="{{ url_for('static', filename='css/singlePageTemplate.css') }}" rel="stylesheet" type="text/css">
9<!--The following script tag downloads a font from the Adobe Edge Web Fonts
   server for use within the web page. We recommend that you do not modify
   it.-->
10<script>var __adobewebfontsappname__="dreamweaver"</script>
11<script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="text/javascript"></script>
12
13<style>
14
15  .hero{
16    background-image: url(static/images/logo1.jpg);
17    background-size: cover; /* For flexibility */
18    height: 290px
19
20  }
21  .banner{
22    height: 160px;
23    margin-top: 70px;
24  }
25</style>
26
27</head>
28<body>
29
30  <!-- Main Container -->
31<div class="container">
32  <!-- Navigation -->
33  <header> <a href="#">
34    <h4 class="logo">MUSK</h4>
35  </a>
```

```
37   <nav>
38     <ul>
39       <li><a href="{{ url_for('index') }}>HOME</a></li>
40       <li><a href="{{ url_for('index') }}>CONTACT</a></li>
41     <li><a href="{{ url_for('index2') }}>LOGIN</a></li>
42   </ul>
43 </nav>
44 </header>
45 <!-- Hero Section -->
46 <section class="hero" id="hero">
47
48   <h2 class="hero_header">MUSK <span class="light">- Personal Assistent</span></h2>
49   <p class="tagline">AN opensource personal assistent INITIATIVE</p>
50 </section>
51 <!-- About Section -->
52 <section class="about" id="about">
53   <h2 class="hidden">About</h2>
54   <p class="text_column">O Musk é um projeto de assistente pessoal de código aberto criado e desenvolvido pelos estudantes da USP Samuel Santos e Guilherme Cabral como elemento avaliativo na disciplina de Projetos de Sistemas Digitais e tem como principal intuito permitir o contato das pessoas com uma tecnologia cada vez mais presente em nosso dia a dia. Fazendo com que o desenvolvimento de projetos de IOTs seja cada vez mais acessível e compreensível.
55   <p class="text_column"><br>Então esse é o Musk um projeto para todos, espero que goste. </p>
56 </section>
57 <!-- Parallax Section -->
58 <section class="banner">
59   <h2 class="parallax">O que o MUSK pode fazer por você?</h2>
60 </section>
61 <!-- More Info Section -->
62 <footer>
63   <article class="footer_column">
64     <h3>MODO VIGIA</h3>
65     
67     <p>Uma das grandes qualidades do MUSK é a de ser os seus olhos e
68       ouvidos quando você
69       não está em casa, mas quer saber como as coisas andam por lá </p>
70   </article>
71   <article class="footer_column">
72     <h3>TEMPERATURA</h3>
73     
73  <p>O MUSK tem tamb m a capacidade de dizer qual a temperatura
     ambiente e a sua humidade, permitindo
     assim que voc sempre possa manter o ambiente confort vel.
     Futuramente o MUSK deve tamb m criar modelos que melhor se adequem
     as suas exigencias e ajustar o ambiente</p>
75  </article>
</footer>
77  <!-- Footer Section -->
<section class="footer_banner" id="contact">
79    <h2 class="hidden">Footer Banner Section </h2>
    <p class="hero_header">PARA MAIS NOT CIAS & ATUALIZA ES</p>
81    <div class="button">subscribe</div>
</section>
83  <!-- Copyrights Section -->
<div class="copyright">&copy;2018 – <strong>Samuel Santos e Guilherme
     Cabral</strong></div>
85 </div>
<!-- Main Container Ends -->
87 </body>
</html>
```

APÊNDICE K – Código HTML Página Aplicativos

```
1  <!doctype html>
2  <html lang="pt-br">
3  <head>
4  <meta charset="utf-8">
5  <meta http-equiv="X-UA-Compatible" content="IE=edge">
6  <meta name="viewport" content="width=device-width, initial-scale=1">
7  <link href="{{ url_for('static', filename='css/simpleGridTemplate.css') }}" 
8      rel="stylesheet" type="text/css">
9
10 <title>MUSK</title>
11 <link href="{{ url_for('static', filename='stylesheets/css/
12     singlePageTemplate.css') }}" rel="stylesheet" type="text/css">
13 <script>var __adobewebfontsappname__="dreamweaver"</script>
14 <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="text/javascript"></script>
15
16 <style>
17     .title_ {
18         grid-template: up;
19     }
20
21 </style>
22
23 </head>
24 <body>
25     <!-- Main Container -->
26     <div class="container">
27         <!-- Navigation -->
28
29         <header id="logo">
30
31             <a class="logo"></a>
32
33             <nav>
34                 <ul>
35                     <li id = "title_musk"><a href="">MUSK</a></li>
36                     <li><a href="{{ url_for('index2') }}">HOME</a></li>
```

```
37      <li> <a href="#">url_for('index')">CONTACT</a></li>
38      <li><a href="#">url_for('index')">LOGOUT</a></li>
39    </ul>
40  </nav>
41</header>
42  <!-- Hero Section -->
43  <section class="intro">
44    <div class="column">
45      <h3>JOHN DOE</h3>
46       </div>
47    <div class="column">
48      <p>Ol , essa a sua p gina principal e aqui estar o disponiveis
49          todos os aplicativos desenvolvidos com o foco de facilitar o seu
50          dia a dia. Futuramente se desenvolver uma loja que permitir
51          que voc baixe e se utilize apenas os aplicativos que voc
52          deseja , mas at l aproveite todos os nossos servi os </p>
53      <p> Lembre-se sempre que estamos de ouvidos abertos a melhorias ,
54          assim caso surja uma nova ideia ou sugest o basta nos contactar.
55          Estamos trabalhando cada vez mais para deixar esse projeto o mais
56          amigavel possivel. </p>
57    </div>
58  </section>
59  <!-- Stats Gallery Section -->
60  <div class="gallery">
61    <div class="thumbnail"> <a href="#">url_for('camera_display')"></a>
63      >
64      <h4>MODO VIGIA</h4>
65      <p class="tag">CMERA</p>
66      <p>Deixe que n s sejamos seus olhos e ouvidos<br> .</p>
67    </div>
68    <div class="thumbnail"> <a href="#">url_for('temp')"></a>
70      <h4>TEMPERATURA</h4>
71      <p class="tag">COMO EST A CASA</p>
72      <p class=> Veja como deixar o seu ambiente mais confortavel</p>
73    </div>
74    <div class="thumbnail"> <a href="#">url_for('agenda')"></a>
76      <h4>AGENDA</h4>
77      <p class="tag">O QUE TEMOS PARA HOJE ?</p>
78      <p> Veja aqui quais s o as suas pr ximas tarefas <br> .</p>
79    </div>
80  </div>
81  <!-- Copyrights Section -->
```

```
73  <div class="copyright">&copy;2018 – <strong>Samuel Santos e Guilherme  
    Cabral</strong></div>  
75 </div>  
75 <!-- Main Container Ends -->  
76 </body>  
77 </html>
```

APÊNDICE L – Controle Motores e Camera HTML

```
1<!doctype html>
2<html>
3<head>
4<meta charset="utf-8">
5<meta http-equiv="X-UA-Compatible" content="IE=edge">
6<meta name="viewport" content="width=device-width, initial-scale=1">
7<title>MUSK</title>
8<link href="{{ url_for('static', filename='css/multiColumnTemplate.css') }}"
9      rel="stylesheet" type="text/css">
10<script>var __adobewebfontsappname__="dreamweaver"</script>
11<script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="text/javascript"></script>
12<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media
13   queries -->
14<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
15<!--[if lt IE 9]>
16    <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js
17      "></script>
18    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></
19      script>
20<![endif]-->
21
22<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js
23      "></script>
24<script language="javascript" type="text/javascript">
25   function Button_onclick(direction) {
26     $.ajax({ url: '/' + direction })
27   }
28</script>
29
30</head>
31<body style="background-color: transparent;">
32<div class="container">
33<header> <a href="">
```

```
35      <li id = "title_musk"><a href="#">{{url_for('index2')}}>MUSK</a></li>
36      <li><a href="#">{{url_for('index2')}}>HOME</a></li>
37      <li><a href="#">{{url_for('contact')}}>CONTACT</a></li>
38      <li><a href="#">{{url_for('index')}}>LOGOUT</a></li>
39
40      </ul>
41  </nav>
42  </header>
43  <section style="background: linear-gradient(to right, lightgray, lightgray)">
44  <p align="right" style="margin: inherit; margin-right: 300px;"><iframe
45      scrolling="no" frameborder="0" src="http://192.168.0.46:8081"></iframe
46      ></p>
47  </section>
48  <div class="row" style="margin-bottom: 400px;">
49      <div class="columns">
50          <p class="thumbnail_align"> <button style="background-color:
51              transparent; border: none;">
52              
54          </button>
55          </p>
56          <h4>UP</h4>
57      </div>
58      <div class="columns">
59          <p class="thumbnail_align"> <button style="background-color:
60              transparent; border: none;">
61              
63          </button> </p>
64          <h4>DOWN</h4>
65      </div>
66      <div class="columns">
67          <p class="thumbnail_align"> <button style="background-color:
68              transparent; border: none;">
69              
71          </button> </p>
72          <h4>LEFT</h4>
73      </div>
74      <div class="columns">
75          <p class="thumbnail_align"> <button style="background-color:
76              transparent; border: none;">
77              
79          </button> </p>
80          <h4>RIGHT</h4>
81      </div>
82  </div>
83
```

```
71 </div>
71   <div class="copyright" style="border: none;">&copy;2018 – <strong>
71     Samuel Santos e Guilherme Cabral</strong></div>
71 </div>
75 </body>
75 </html>
```

APÊNDICE M – Código HTML Temperatura e Umidade

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0,
7              maximum-scale=1">
8
9      <title>MUSK</title>
10
11     <!-- Loading third party fonts -->
12     <link href="http://fonts.googleapis.com/css?family=Roboto:300,400,700|"
13         rel="stylesheet" type="text/css">
14     <link href="fonts/fontawesome.min.css" rel="stylesheet" type="text/css"
15         ">
16
17     <!-- Loading main css file -->
18     <link rel="stylesheet" href="{{ url_for('static', filename='css/
19         temp_style.css') }}>
20
21     <!--[if lt IE 9]>
22     <script src="js/ie-support/html5.js"></script>
23     <script src="js/ie-support/respond.js"></script>
24     <![endif]-->
25
26     <script>var __adobewebfontsappname__="dreamweaver"</script>
27 <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type=""
28         text/javascript"></script>
29
30     </head>
31
32     <body>
33         <!--style="background-color: lightgray"-->
34         <div class="site-content" style="background-color: white;">
35             <header> <a href="">
36                 <h4 class="logo" style="margin: inherit; margin-left: 30px; font-
37                     family: 'source-sans-pro';">MUSK</h4>
38             </a>
39
40             <!--content-->
41         </div>
42
43         <!--script-->
44
45     </body>
46
47 </html>
```

```
35      <nav>
36          <ul>
37              <li><a style="font-family: 'source-sans-pro';" href="{{ url_for('
38                  index2') }}">HOME</a></li>
39              <li><a style="font-family: 'source-sans-pro';" href="{{ url_for('
40                  contact') }}">CONTACT</a></li>
41              <li><a style="font-family: 'source-sans-pro';" href="{{ url_for('
42                  index') }}">LOGOUT</a></li>
43          </ul>
44      </nav>
45      </header>
46
47      <!-- .site-header -->
48
49
50      <div class="forecast-table" style="background-color: white;">
51          <div class="container" style="margin-top: 190px;">
52
53              <div class="forecast-container">
54                  <div class="today-forecast">
55                      <div class="forecast-header">
56                          <div class="day" style="float: none;">Temperatura – Sala</
57                          div>
58
59                      </div> <!-- .forecast-header -->
60                      <div class="forecast-content">
61                          <div class="degree">
62                              <div class="num">{{ temp }}<sup>0</sup>C</div>
63                              <div class="forecast-icon">
64                                  </div>
66                          </div>
67                          </div>
68
69                      </div>
70
71                  <div class="forecast">
72                      <div class="forecast-header">
73                          <div class="day"> Umidade</div>
74                      </div> <!-- .forecast-header -->
75                      <div class="forecast-content">
76                          <div class="num" style="font-size: 62px;">{{ umid }} %</div>
77
78                          <div class="forecast-icon">
79                              
81                          </div>
82
83                      </div>
84
85                  </div>
86
87
88
89
90
91
92
93
```

```
75          </div>
76      </div>
77  </div>
78</div>
79<main class="main-content">
80  <div class="fullwidth-block" style="margin-bottom: inherit;
81    background-color: #52bad5;">
82    <div class="container">
83      <h2 class="section-title">Controle</h2>
84      <div class="row">
85        <div class="col-md-3 col-sm-6">
86          <div class="live-camera">
87            <figure class="live-camera-cover" style="border: none;">
88              </figure>
90            <h3 align="center" class="location">Ar-Condicionado - 1</
91              h3>
92
93          </div>
94        </div>
95        <div class="col-md-3 col-sm-6">
96          <div class="live-camera">
97            <figure class="live-camera-cover" style="border: none;">
98              </figure>
99            <h3 align="center" class="location">Ventilador - Teto 1</
100           h3>
101
102          </div>
103        </div>
104        <div class="col-md-3 col-sm-6">
105          <div class="live-camera">
106            <figure class="live-camera-cover" style="border: none; "
107              ></figure>
109            <h3 align="center" class="location">Televisor</h3>
110
111          </div>
112        </div>
113        <div class="col-md-3 col-sm-6">
114          <div class="live-camera">
115            <figure class="live-camera-cover" style="border: none;">
116              </figure>
117            <h3 align="center" class="location">SoundBar</h3>
118
119          </div>
120        </div>
121      </div>
122    </div>
123  </div>
124</main>
```

```
113         </div>
114     </div>
115 </div>
116
117 <div class="fullwidth-block">
118     <div class="container">
119         <h2 class="section-title" style="color: #A3A3A3;">C modos</h2>
120         <div class="row">
121             <div class="col-md-3 col-sm-6">
122                 <div class="live-camera">
123                     <figure class="live-camera-cover"></figure>
124                 <h3 class="location" style="color: #A3A3A3;">Quarto do
125                     B b </h3>
126                 </div>
127             </div>
128             <div class="col-md-3 col-sm-6">
129                 <div class="live-camera">
130                     <figure class="live-camera-cover"></figure>
131                     <h3 class="location" style="color: #A3A3A3;">Sala</h3>
132                 </div>
133             </div>
134             <div class="col-md-3 col-sm-6">
135                 <div class="live-camera">
136                     <figure class="live-camera-cover"></figure>
137                     <h3 class="location" style="color: #A3A3A3;">Quarto
138                         Principal</h3>
139                     </div>
140                 </div>
141             <div class="col-md-3 col-sm-6">
142                 <div class="live-camera">
143                     <figure class="live-camera-cover"></figure>
144                     <h3 class="location" style="color: #A3A3A3;">Cozinha</h3>
145                 </div>
146             </div>
147         </div> <!-- .main-content -->
148
149 <div class="copyright"> © 2018 - <strong>Samuel Santos e Guilherme
150 Cabral</strong> </div>
```

```
151  <!-- .site-footer -->
151  </div>

153  <script src="js/jquery-1.11.1.min.js"></script>
153  <script src="js/plugins.js"></script>
155  <script src="js/app.js"></script>

157  </body>

159 </html>
```

APÊNDICE N – Todo HTML

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta name="description" content="">
8   <meta name="author" content="">
9
10  <title>MUSK</title>
11
12  <!-- css -->
13  <link href="{{ url_for('static', filename='css/bootstrap.min.css') }}" rel="stylesheet" type="text/css">
14  <link href="{{ url_for('static', filename='font-awesome/css/font-awesome.min.css') }}" rel="stylesheet" type="text/css" />
15  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='plugins/cubeportfolio/css/cubeportfolio.min.css') }}>
16  <link href="{{ url_for('static', filename='ss/nivo-lightbox.css') }}" rel="stylesheet" />
17  <link href="{{ url_for('static', filename='css/nivo-lightbox-theme/default/default.css') }}" rel="stylesheet" type="text/css" />
18  <link href="{{ url_for('static', filename='css/owl.carousel.css') }}" rel="stylesheet" media="screen" />
19  <link href="{{ url_for('static', filename='css/owl.theme.css') }}" rel="stylesheet" media="screen" />
20  <link href="{{ url_for('static', filename='css/animate.css') }}" rel="stylesheet" />
21  <link href="{{ url_for('static', filename='css/todo_style.css') }}" rel="stylesheet">
22
23  <!-- boxed bg -->
24  <link id="bodybg" href="bodybg/bg1.css" rel="stylesheet" type="text/css" />
25  <!-- template skin -->
26  <link id="t-colors" href="{{ url_for('static', filename='color/default.css') }}" rel="stylesheet">
27
28  <script>var __adobewebfontsappname__="dreamweaver"</script>
29  <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="text/javascript"></script>
```

```
31 </head>
33 <body id="page-top" data-spy="scroll" data-target=".navbar-custom">
35 <div id="wrapper">
37 <header> <a href="">
38   <h4 class="logo" style="font-family: 'source-sans-pro'; font-size: 15px; margin: inherit; margin-top: 20px; margin-left: 30px;">MUSK</h4>
39 </a>
40 <nav style="margin-top: 10px;">
41   <ul>
42     <li><a href="{{url_for('index2')}}" style="font-family: 'source-sans-pro'; font-size: 15px;">HOME</a></li>
43     <li><a href="{{url_for('contact')}}" style="font-family: 'source-sans-pro'; font-size: 15px;">CONTACT</a></li>
44     <li><a href="{{url_for('index')}}" style="font-family: 'source-sans-pro'; font-size: 15px;">LOGOUT</a></li>
45   </ul>
46 </nav>
47 </header>
48 <header>&nbsp;<p></p></header>
49
50
51 <!-- Section: intro -->
52 <section id="intro" class="intro">
53   <div class="intro-content">
54     <div class="container">
55       <div class="row">
56         <div class="col-lg-6">
57           <div class="wow fadeInDown" data-wow-offset="0" data-wow-delay="0.1s">
58             <h2 class="h-ultra">Agende Suas Tarefas</h2>
59           </div>
60           <div class="wow fadeInUp" data-wow-offset="0" data-wow-delay="0.1s">
61             <h4 class="h-light"><p></p>Tarefas a <span class="color" style="color: #931C1E;">Finalizar:</span></h4>
62           </div>
63           <div class="well well-trans">
64             <div class="wow fadeInRight" data-wow-delay="0.1s">
65               <ul class="lead-list">
```

```

67
68  {% for todo in incomplete %}
69      <li><span><a href="#">{{ url_for('complete', id=todo.id) }}>
70          </a> </span>
72          <span class="list"><strong>{{ todo.title }}, {{ todo.start
73              }}</strong></span>
74      </li>
75  {% endfor %}
76
77
78
79      </div>
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105

```

```
107         <div class="row">
108             <div class="col-xs-6 col-sm-6 col-md-6">
109                 <div class="form-group">
110                     <label style="color: white;">.</label>
111                 </div>
112             <div class="col-xs-6 col-sm-6 col-md-6">
113                 <div class="form-group">
114                     <label style="color: white;">.</label>
115                 </div>
116             </div>
117         </div>
118
119         <input type="submit" value="Submit" class="btn btn-skin
120             btn-block btn-lg">
121         </form>
122     </div>
123 </div>
124
125     </div>
126     </div>
127 </div>
128
129     <div class="row">
130         <div class="col-lg-6" style="margin-right: 300px;">
131             <div class="wow fadeInDown" data-wow-offset="0" data-wow-delay="0.1s">
132                 <h2 class="h-ultra"></h2>
133             </div>
134             <div class="wow fadeInUp" data-wow-offset="0" data-wow-delay="0.1s">
135                 <h4 class="h-light">Tarefas <span class="color">Finalizadas:</span></h4>
136             </div>
137             <div class="well well-trans">
138                 <div class="wow fadeInRight" data-wow-delay="0.1s">
139
140                     <ul class="lead-list">
141                         {% for todo in complete %}
142                             <li><span class="list"><
144                                 strong>{{ todo.title }}</strong></span></li>
145                         {% endfor %}
146                     </ul>
```

```
145      </div>
146      </div>
147      </div>
148      </div>
149      </div>
150      </div>
151      </section>
152
153  </div>
154  <div class="copyright" style="font-family: 'source-sans-pro';">&copy;
155  ;2018 - <strong>Samuel Santos e Guilherme Cabral</strong></div>
156
157 <a href="#" class="scrollup"><i class="fa fa-angle-up active"></i></a>
158
159  <!-- Core JavaScript Files -->
160  <script src="js/jquery.min.js"></script>
161  <script src="js/bootstrap.min.js"></script>
162  <script src="js/jquery.easing.min.js"></script>
163  <script src="js/wow.min.js"></script>
164  <script src="js/jquery.scrollTo.js"></script>
165  <script src="js/jquery.appear.js"></script>
166  <script src="js/stellar.js"></script>
167  <script src="plugins/cubeportfolio/js/jquery.cubeportfolio.min.js"></
168      script>
169  <script src="js/owl.carousel.min.js"></script>
170  <script src="js/nivo-lightbox.min.js"></script>
171  <script src="js/custom.js"></script>
172
173 </body>
174
175 </html>
```