

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, n^o. USP: 9403343

Autor: Samuel S. do Espírito Santo, n^o. 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

Resumo

Este projeto busca desenvolver um assistente pessoal de baixo custo que permita a democratização do conhecimento e da tecnologia. Tendo como principal elemento uma Raspberry Pi 3, o assistente pessoal MUSK busca servir como um projeto base para desenvolvimentos futuros, contando com algumas funções como agenda, câmera de segurança, detecção de presença e apresentação de temperatura e umidade.

O MUSK tem também 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.

Lista de ilustrações

Figura 1 – Raspberry Pi 3 Model B	14
Figura 2 – Display LCD 3.5inch	16
Figura 3 – Sensor de presença PIR HC-SR501	18
Figura 4 – Hardware PIR	19
Figura 5 – Ajustes de operação do PIR	19
Figura 6 – Sensor de temperatura DHT11	20
Figura 7 – WebCam Goldship 3817	20
Figura 8 – Super Mini USB 2.0 Microphone	21
Figura 9 – Servo motor TowerPro MG995	22
Figura 10 – PCA9685 Adafruit	23
Figura 11 – Disposição do sistema operacional no computador	24
Figura 12 – Estrutura HTML	27
Figura 13 – Estrutura HTML (1)	28
Figura 14 – Estrutura HTML (2)	28
Figura 15 – Estrutura HTML (3)	28
Figura 16 – Estrutura HTML com componente de javascript	29
Figura 17 – Arquitetura do Sistema	36
Figura 18 – Cartão micro SD	36
Figura 19 – Configurações no MAC	38
Figura 20 – Terminal Putty	39
Figura 21 – Interface DreamWeaver	41
Figura 22 – Função Ajax para o Controle dos servos Motores	42
Figura 23 – Código HTML, CSS - Temperatura e Umidade	42
Figura 24 – Código Python - Temperatura e Umidade	42
Figura 25 – Código HTML – complete, Incomplete	43
Figura 26 – Método de Tarefa Completada	44
Figura 27 – Exemplo arquivo Python	45
Figura 28 – Exemplo arquivo kivy	46
Figura 29 – Tela kivy	47
Figura 30 – Exemplo código Python com kivy	48
Figura 31 – Interface Kivy (1)	49
Figura 32 – Código Kivy - Interface (1)	49
Figura 33 – Código Kivy - Interface (2)	50
Figura 34 – Kivy - Interface (2)	51
Figura 35 – Código Kivy - Interface (3)	52
Figura 36 – Kivy - Interface (3)	53

Figura 37 – Esquemático motores e RPi	54
Figura 38 – Rotas de controle do motor	55
Figura 39 – Método <i>servo_position</i>	55
Figura 40 – Esquemático PIR	56
Figura 41 – Método de configuração do sensor PIR	57
Figura 42 – Método de envio do email	57
Figura 43 – Esquemático DHT11	58
Figura 44 – Método <i>readDht11</i>	58
Figura 45 – Rota para receber os valores do sensor DHT11	59
Figura 46 – Menu de Configuração RPi, Opções Avançadas	59
Figura 47 – Menu de Configuração RPi, Expandir Sistema	60
Figura 48 – Método teste OpenCV	64
Figura 49 – Método <i>runFaceDetector</i>	64
Figura 50 – Método <i>getImage</i>	65
Figura 51 – Método <i>sendVideo</i>	65
Figura 52 – Método <i>generateVideo, server.py</i>	66
Figura 53 – Endereço <i>cameraview</i>	66
Figura 54 – Função <i>create_collection</i>	68
Figura 55 – Função <i>index_face</i>	68
Figura 56 – Configuração do arquivo <i>config.txt</i>	69
Figura 57 – Configuração do arquivo <i>asounf.conf</i>	69
Figura 58 – Banco de dados python (1)	72
Figura 59 – Banco de dados python (2)	72
Figura 60 – Página de Entrada WEB	74
Figura 61 – Página de Aplicativos	75
Figura 62 – Exemplo Interface Modo Vigia	76
Figura 63 – Interface de Temperatura e Umidade	77
Figura 64 – Interface Agenda	78
Figura 65 – Teste da Variação do Tamanho da Imagem Enviada	80
Figura 66 – Picos de tempo no início do teste	81
Figura 67 – Teste com variação da imagem de entrada	82
Figura 68 – Teste em tempo real	83
Figura 69 – Teste de temperatura	85

Lista de tabelas

Tabela 1 – Características técnicas Display LCD 3.5inch	15
Tabela 2 – Características Display LCD	17
Tabela 3 – Características técnicas PIR	18
Tabela 4 – Características técnicas DHT11	19
Tabela 5 – Características técnicas WebCam Goldship 3817	21
Tabela 6 – Características técnicas Super Mini USB 2.0 Microphone	21
Tabela 7 – Características técnicas TowerPro MG995	22

Sumário

	Lista de ilustrações	4
	Lista de tabelas	6
	Sumário	7
1	INTRODUÇÃO	10
1.1	Objetivo	10
1.2	Organização do Trabalho	11
2	CONCEITOS E FUNDAMENTOS	13
2.1	Produto Final	13
2.1.1	Sistemas Embarcados	13
2.2	Estrutura de Hardware	13
2.2.1	Raspberry Pi 3 Model B	14
2.2.2	LCD	15
2.2.3	Sensores	17
2.2.3.1	Sensor de Presença	18
2.2.3.2	Sensor de Temperatura e Umidade	19
2.2.4	Câmera	20
2.2.5	Mini Microfone USB	21
2.2.6	Servo Motor	21
2.3	Sistemas Operacionais	23
2.3.1	Raspbian	24
2.4	Estrutura de Software	25
2.4.1	Python	25
2.4.2	Web Server	25
2.4.2.1	Flask	26
2.4.3	Páginas WEB	26
2.4.3.1	HTML	27
2.4.3.2	CSS	27
2.4.3.3	JavaScript	28
2.4.4	Banco de Dados	29
2.4.4.1	SQLite	30
2.4.5	Interface Homem-Máquina	31
2.4.5.1	Interface Gráfica	31

2.4.5.2	Visão Computacional	31
2.4.5.2.1	OpenCV	32
2.4.5.3	Amazon Web Services	33
2.4.5.3.1	AWS Rekognition	33
2.4.5.4	Comando de Voz	34
2.4.5.4.1	Google Assistant	35
3	DESENVOLVIMENTO	36
3.1	Arquitetura do Sistema	36
3.2	Configuração da Raspberry PI	36
3.3	Configuração do Servidor Flask	39
3.4	Web	40
3.4.1	DreamWeaver	40
3.4.2	Página Inicial e de Aplicativos	41
3.4.3	Página Modo Vigia	41
3.4.4	Página Sensoriamento (Temperatura e Umidade)	42
3.4.5	Página Agenda	43
3.5	Interface Gráfica	44
3.5.0.1	Geração de Telas	48
3.6	Sistema de Segurança	53
3.6.1	Configuração do Motor	53
3.6.2	Configuração do Sensor PIR	55
3.6.3	Configuração DHT11	57
3.6.4	Configuração OpenCV	59
3.6.4.1	Preparando Ambiente OpenCV	63
3.6.4.2	Implementação Geral	64
3.6.4.3	Implementação WEB	65
3.6.5	Configuração AWS Rekognition	66
3.7	Comando de Voz	69
3.7.1	Configuração do Google Assistant	69
3.7.2	Desenvolvimento de Skills	71
3.8	Desenvolvimento do Banco de Dados	71
4	RESULTADOS E DISCUSSÕES	73
4.1	Interação com o Usuário	73
4.1.1	Web	73
4.1.1.1	Página Inicial	73
4.1.1.2	Página de Aplicativos	74
4.1.1.3	Página Modo Vigia	75
4.1.1.4	Sensor de Temperatura e Umidade	76

4.1.1.5	Agenda WEB	77
4.2	Sistema de Segurança	78
4.2.1	Controle Câmera	78
4.2.2	Detector de Presença	79
4.2.3	Detecção e Reconhecimento Facial	79
4.2.3.1	Testes Detecção de Faces	80
4.2.3.2	Testes AWS Rekognition	80
4.3	Controle de Voz	83
4.4	Performance do Sistema	83
4.5	Aprimoramentos Futuros	85
5	CONCLUSÃO	86
	Referências	87
	 APÊNDICES	 93
	APÊNDICE A – CÓDIGO PÁGINA PRINCIPAL HTML	94
	APÊNDICE B – CÓDIGO HTML PÁGINA APLICATIVOS	97
	APÊNDICE C – CONTROLE MOTORES E CAMERA HTML	100
	APÊNDICE D – CÓDIGO HTML TEMPERATURA E UMIDADE	103
	APÊNDICE E – PROGRAMA PRINCIPAL EM PYTHON	108
	APÊNDICE F – TODO HTML	118
	APÊNDICE G – TODO PYTHON	123

1 Introdução

A humanidade, com o passar dos anos, vem encontrando novas formas cada vez mais orgânicas de se utilizar da tecnologia no auxílio 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 estas, poucas acabam sendo acessíveis ao publico 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 auxílio da tecnologia. Motivado por essa nova tendência que nasce a ideia do assistente pessoal apresentado neste trabalho o qual recebeu o nome de !!!!. Um assistente pessoal de código aberto que permite acesso a tecnologias de ponta, tornando dessa forma o conhecimento deste tipo de aplicação muito mais democrático.

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.

Com o intuito de verificar o funcionamento do assistente pessoal aqui projetado será construído um protótipo. Por sua vez o protótipo é uma iniciativa que serve como base para o desenvolvimento de projetos mais complexos e diversificados.

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 a 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 – senão o primeiro – projeto 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 acessíveis 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 esta 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 Apps e ter o seu assistente pessoal personalizado, desta forma popularizando 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 Conceitos e 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.

2.1.1 Sistemas Embarcados

Segundo a definição de Stallings, um sistema embarcado consiste em um sistema microcontrolado 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 performance.

2.2 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.

2.2.1 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 uma performance comparável com a de um desktop 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, 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: (9)

- 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;

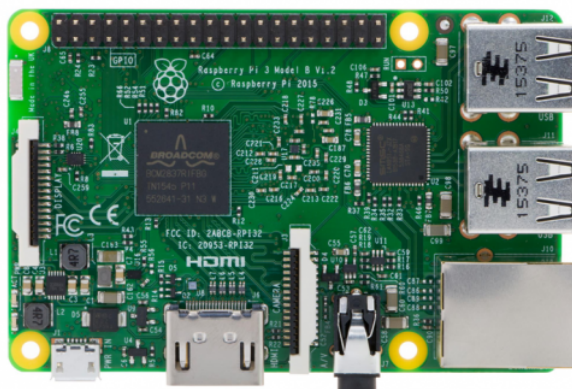


Figura 1 – Raspberry Pi 3 Model B

A versão três da RPi é comparada, por muitos, a um PC devido a sua alta performance (10). Esse upgrade em sua qualidade 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 (11):

- Representação 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.

2.2.2 LCD

Para que fosse possível a interação visual com o usuário instalou-se uma tela LCD sensível(12)(13). A tela LCD conta com 26 pinos, sendo que 9 deles são NC - ou seja, Não Conectados - permitindo assim a conexão de outros sensores e atuadores. As características da tela LCD são apresentadas na tabela (1).

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

Tabela 1 – Características técnicas Display LCD 3.5inch

A tela instalada está presente na figura (2).

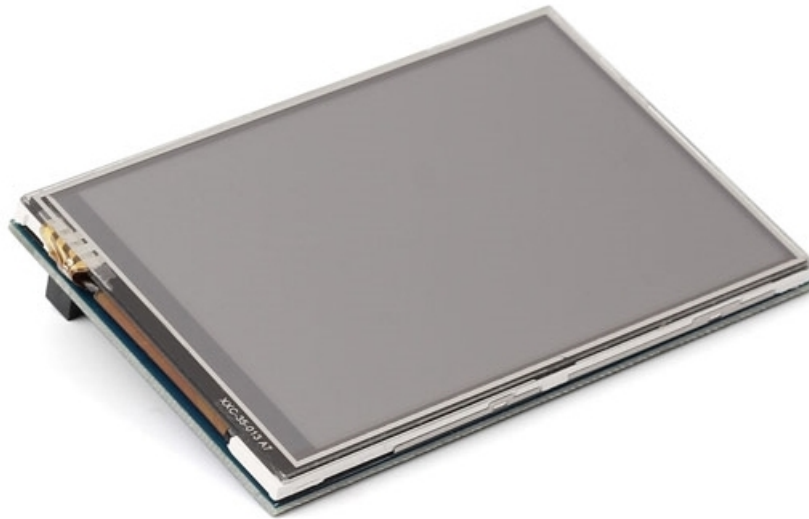


Figura 2 – Display LCD 3.5inch

Por fim os pinos de saída estão presentes na tabela (2).

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	$TP_I RQ$	<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

Tabela 2 – Características Display LCD

2.2.3 Sensores

O MUSK utiliza sensores para detectar e responder estímulos do ambiente em que está posicionado, como: detecção de presença e variações de temperatura. Um sensor é definido como um componente eletrônico que tem a capacidade de transformar um evento físico em informação elétrica, ou seja, tensão (14). 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 (15).

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 como zero ou um (15).

2.2.3.1 Sensor de Presença

O sensor responsável pela verificação de eventuais presenças no ambiente monitorado é o PIR HC-SR501, que foi selecionado, uma vez que ele é um dos mais utilizados devido a sua qualidade e facilidade de aplicação. A seguir estão representadas na tabela as características do sensor. (16)

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

Tabela 3 – Características técnicas PIR



Figura 3 – Sensor de presença PIR HC-SR501

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.

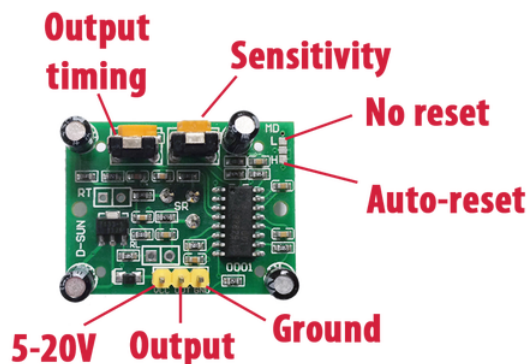


Figura 4 – Hardware PIR

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

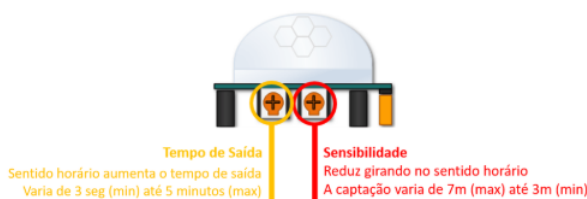


Figura 5 – Ajustes de operação do PIR

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.

2.2.3.2 Sensor de Temperatura e Umidade

No projeto é utilizado o sensor DHT11 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 a seguir. (17)

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

Tabela 4 – Características técnicas DHT11

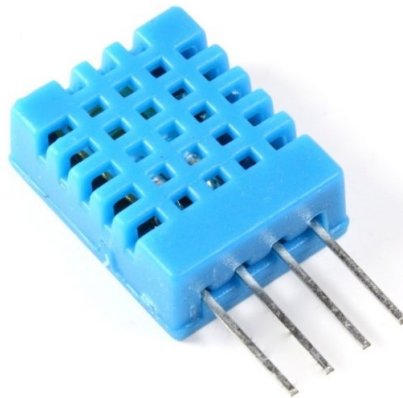


Figura 6 – Sensor de temperatura DHT11

2.2.4 Câmera

A câmera utilizada para a obtenção e *streaming* de imagens foi a WebCam modelo Goldship 3817. A qual conta com uma resolução de 1.3M pixels e foi streamada em uma janela de 600 por 420 (18). A utilização de uma Webcam com entrada USB comum, apesar de gerar certo atraso em relação a uma Webcam 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 webcam 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

As características do modelo selecionado estão listadas a seguir.

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

Tabela 5 – Características técnicas WebCam Goldship 3817

2.2.5 Mini Microfone USB

Para a integração do serviço de comando de voz foi necessário utilizar o Super Mini USB 2.0 Microphone. 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 (19). Apesar disso, a API do Google Assistant consegue absorver as informações do áudio de forma eficiente.



Figura 8 – Super Mini USB 2.0 Microphone

As especificações estão listadas na tabela a baixo.

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

Tabela 6 – Características técnicas Super Mini USB 2.0 Microphone

2.2.6 Servo Motor

Para o controle posicional do Musk foi necessário a utilização do Servo TowerPro MG995, uma vez que ele possui uma alta qualidade, precisão, dimensão e facilidade no

controle. As características principais estão sendo amostrados na tabela a seguir.

Parâmetro	Valor
Tensão de Operação	4,8 a 7,2V
Modulação	Analógica
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

Tabela 7 – Características técnicas TowerPro MG995



Figura 9 – Servo motor TowerPro MG995

O servo motor é um dispositivo eletromecânico controlado a partir da Modulação da Largura de Pulso (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 de baixo custo que estabelece a comunicação dos servos com a RPi através do protocolo (20).

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.

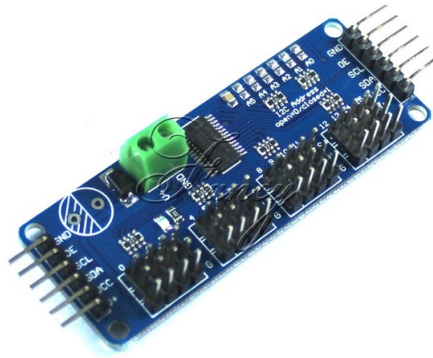


Figura 10 – PCA9685 Adafruit

2.3 Sistemas Operacionais

A definição de sistema operacional (SO) é algo um pouco impreciso (21). 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 (21):

- **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 (21).

A figura a seguir ilustra de forma clara a disposição em camadas do sistema operacional dentro do computador.

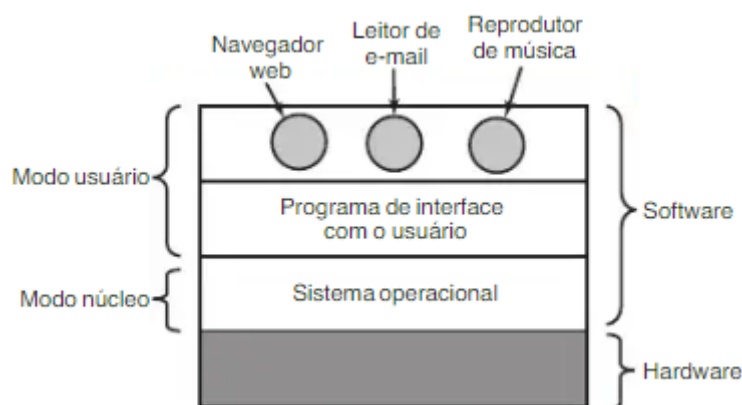


Figura 11 – Disposição do sistema operacional no computador

2.3.1 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 (9).

Este sistema operacional é fornecido gratuitamente, sendo inicialmente desenvolvido por Mike Thompson e Peter Green juntamente com entusiastas da placa Raspberry Pi (22). O Raspbian tem como o seu ambiente de desktop o LXDE, como gerenciador de janelas o OpenBox e navegador Chromium. Disponibilizado pela fundação Raspberry Pi este SO conta com uma versão do Mathematica e de uma versão para a Rasperry 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. (22) (23)

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.

2.4 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.

2.4.1 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 capacidade de 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. (24)

2.4.2 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 (25).

Os principais métodos do protocolo HTTP estão descritos a seguir (26).

- **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.

2.4.2.1 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 (27).

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 (28).

2.4.3 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) (29)(30). 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 uteis 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.

2.4.3.1 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 (31)(32). 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.: < >).

A seguir 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”.

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

Figura 12 – Estrutura HTML

A *tag html* delimita o início e o fim dos códigos em HTML. A *tag head* é responsável por “linkar”, organizar e apresentar certos arquivos, funções e informações que não são visíveis diretamente na página (31). 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 (31).

2.4.3.2 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(33).

Existem algumas maneiras diferentes de se utilizar o CSS ao longo de uma página HTML. A mais 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* (33). Um exemplo pode ser observado a seguir.

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

Figura 13 – Estrutura HTML (1)

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

Figura 14 – Estrutura HTML (2)

Dentro do arquivo .css introduzido no *head* tem-se o seguinte modelo.

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

Figura 15 – Estrutura HTML (3)

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.

2.4.3.3 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 então abre espaço para scripts os quais funcionam de maneira muito semelhante a 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 a seguir.

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

Figura 16 – Estrutura HTML com componente de javascript

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.

2.4.4 Banco de Dados

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

"Uma coleção de dados inter-relacionados, representando informações sobre um domínio específico." (34)

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." (35)

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(36).

- **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(37).

2.4.4.1 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 quantidades 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. (38)

2.4.5 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.

2.4.5.1 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 orgânica HMI. Por tanto utilizou-se 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 modernas, 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.

2.4.5.2 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 compreensível para o computador (39).

Entre as tarefas típicas de visão computacional, pode-se citar (40):

- 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 (41).

2.4.5.2.1 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(42)(43). Ao longo dos anos o OpenCV contou com varias 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 sua funções mais evidentes são (43):

- **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.

2.4.5.3 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 (44):

- **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:** alta performance no processamento e transmissão de dados, tolerância a falhas;
- **Zonas de Disponibilidade:** são Datacenters separados e isolados fisicamente, que são conectados com baixa latência, alto throughput 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;

2.4.5.3.1 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 (45).

- **Rótulos:** objetos, eventos, conceitos (noite, natureza) e atividades;

- **Faces:** detecta faces em imagens/vídeo, extraindo 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.

2.4.5.4 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 (46).

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.

2.4.5.4.1 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 (47).

3 Desenvolvimento

3.1 Arquitetura do Sistema

A figura a baixo ilustra o esquemático da arquitetura do sistema.

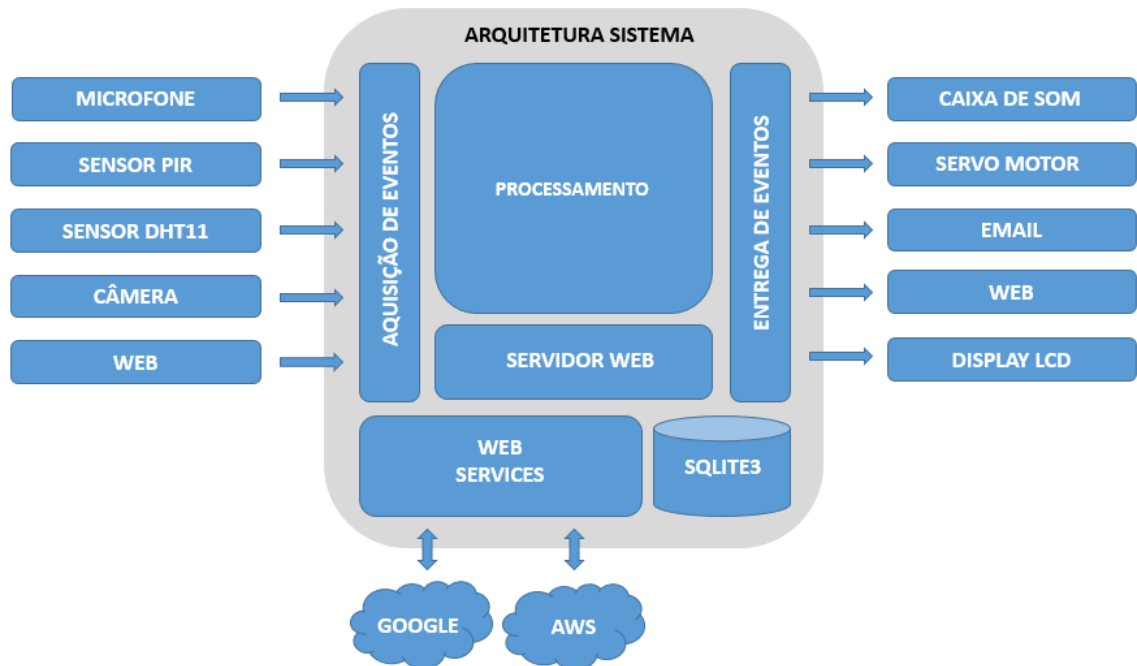


Figura 17 – Arquitetura do Sistema

3.2 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.



Figura 18 – Cartão micro SD

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. (48) (49)

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:

```
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-default.link
```

Determinou-se o MAC adress desejado a partir do spoofedMAC: (50)

```
[Match]
MACAddress=original MAC

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

Figura 19 – Configurações no MAC

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: (51)

```
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 1.

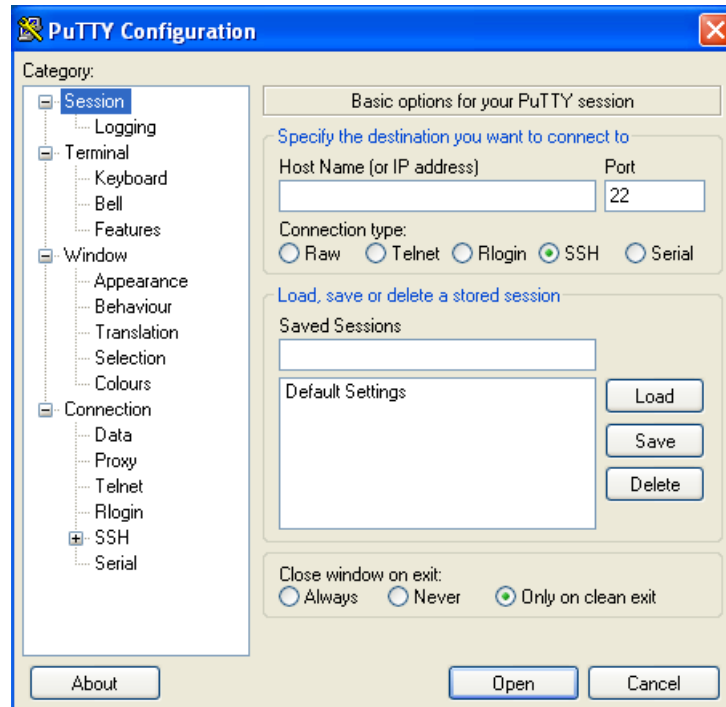


Figura 20 – Terminal Putty

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.

3.3 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. (52)

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.

3.4 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 liscença de 15 dias gratuitos de teste. Ao longo deste tópico será abordado o desenvolvimento da interface WEB a partir do DreamWeaver.

3.4.1 DreamWeaver

O DreamWeaver é uma plataforma de desenvolvimento web baseado em WY-SIWYG (*What You See Is What You Get* – O Que Você Vê É O Que Você Têm, em português) que possui suporte varias tecnologias WEB, como HTML, XHTML, CSS, JavaScript, Ajax, PHP, ASP, ASP.NET, JSP, entre outros (53).

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 paginas web. Pode-se observar a sua interface na figura 21.

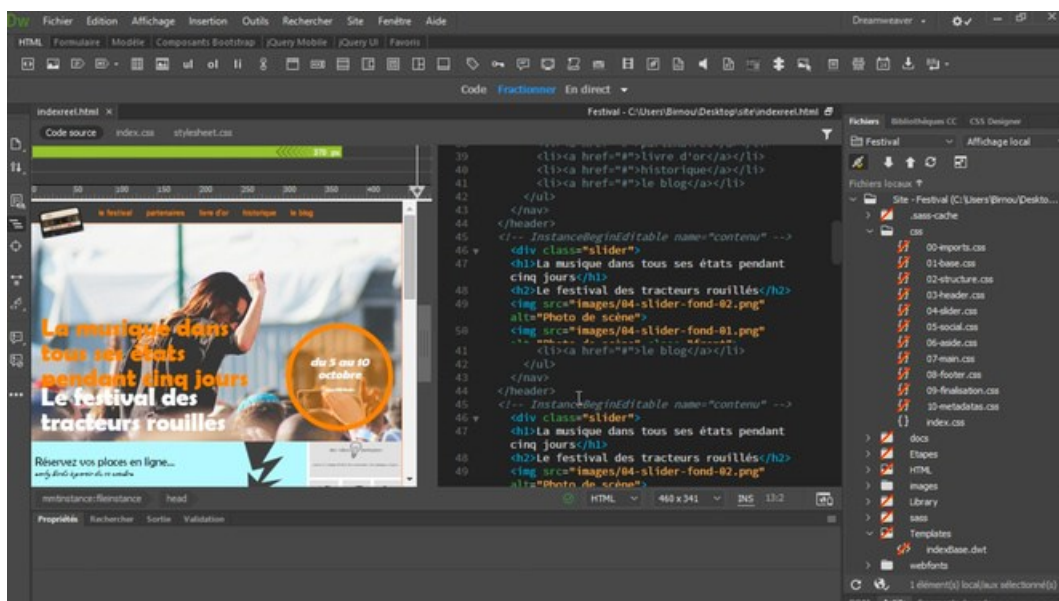


Figura 21 – Interface DreamWeaver

3.4.2 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 esta descrito no apêndice A, o resultado por sua vez pode ser observado na figura 60.

3.4.3 Página Modo Vigia

A concepção desta pagina 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 comodo, mas também mova a câmera.

Sobre essa interface funciona um dos elementos chaves para o controle dos servo 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.


```

<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>

```

Figura 22 – Função Ajax para o Controle dos servos Motores

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 servo motores devem seguir, conforme visto no apêndice C. Onde pode se observar cada botão envia uma “string” para a função que altera a URL de acordo com o botão acionado.

3.4.4 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.

```

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

```

Figura 23 – Código HTML, CSS - Temperatura e Umidade

Como pode ser observado no código do apêndice D 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. (54)

```

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

```

Figura 24 – Código Python - Temperatura e Umidade

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

3.4.5 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 F - ao arquivo em python para que o mesmo salve as atividades em um banco de dados.

Como pode ser visto no apêndice F 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. ((55))

Como pode ser observado no apêndice G 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. ((56))

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 a 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.).

```
<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
    {% endfor %}
  </ul>
  <h2>Completed Items</h2>
  <ul>
    {% for todo in complete %}
    <li style="font-size: 30pt">{{ todo.text }}</li>
    {% endfor %}
  </ul>
</div>
```

Figura 25 – Código HTML – complete, Incomplete

Já no código em python 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.

```
@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'))
```

Figura 26 – Método de Tarefa Completada

3.5 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 a baixo:

```
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-enchanted 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 do exemplo a seguir.

```
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()
```

Figura 27 – Exemplo arquivo Python

```
# ----- FLOATING.KV -----

# size_hint defines the widget width (0-1) representing
# a percentage of 100% and height of the window
<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

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

<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}
```

Figura 28 – Exemplo arquivo kivy

Conforme pode ser visto na figura (27) após ser acionado o arquivo Python busca instancias 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)

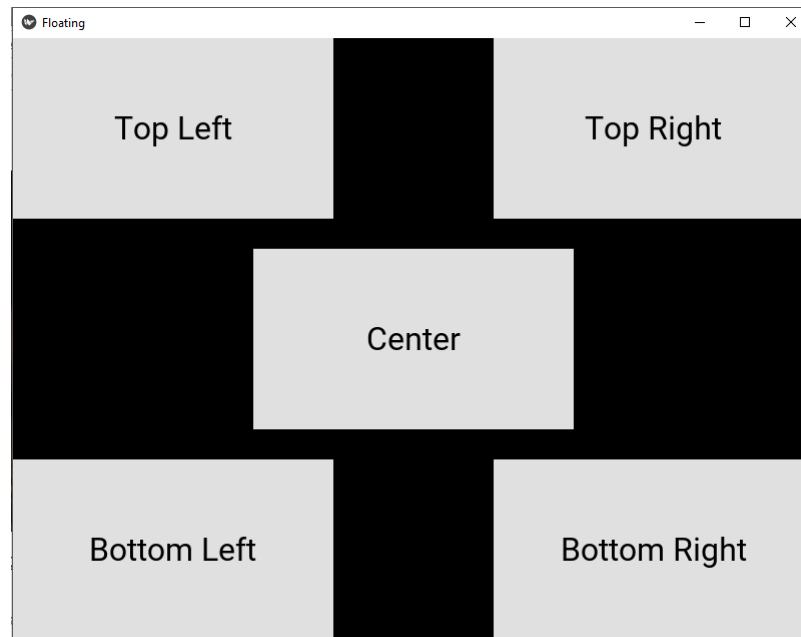


Figura 29 – Tela kivy

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.

```

# ----- 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()

flApp = FloatingApp()
flApp.run()

```

Figura 30 – Exemplo código Python com kivy

3.5.0.1 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 E. 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.

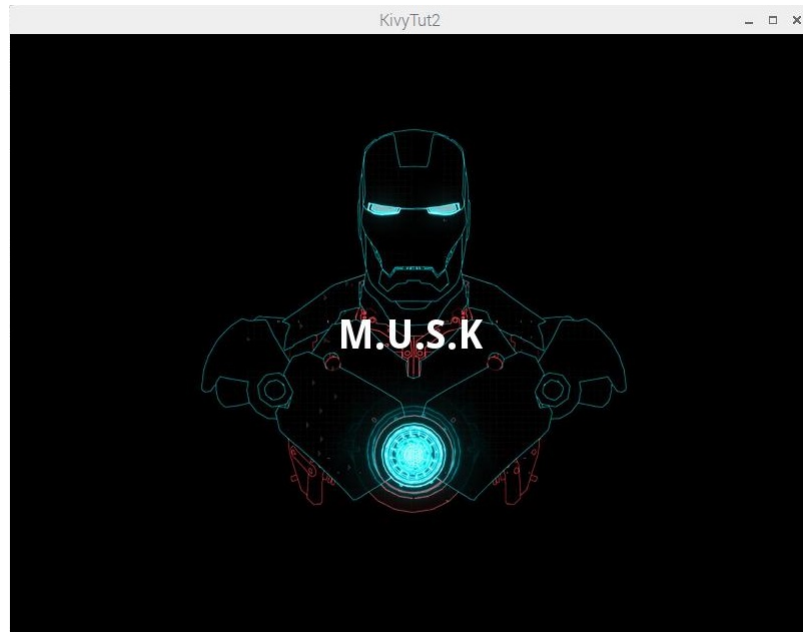


Figura 31 – Interface Kivy (1)

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

```
<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
```

Figura 32 – Código Kivy - Interface (1)

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 dítório 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 default 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).

```
<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}
```

Figura 33 – Código Kivy - Interface (2)

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



Figura 34 – Kivy - Interface (2)

A tela de dados internos por sua vez é 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. O código utilizado para gerar tal tela pode ser observado na figura (35).



Figura 36 – Kivy - Interface (3)

3.6 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.

3.6.1 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 a seguir.

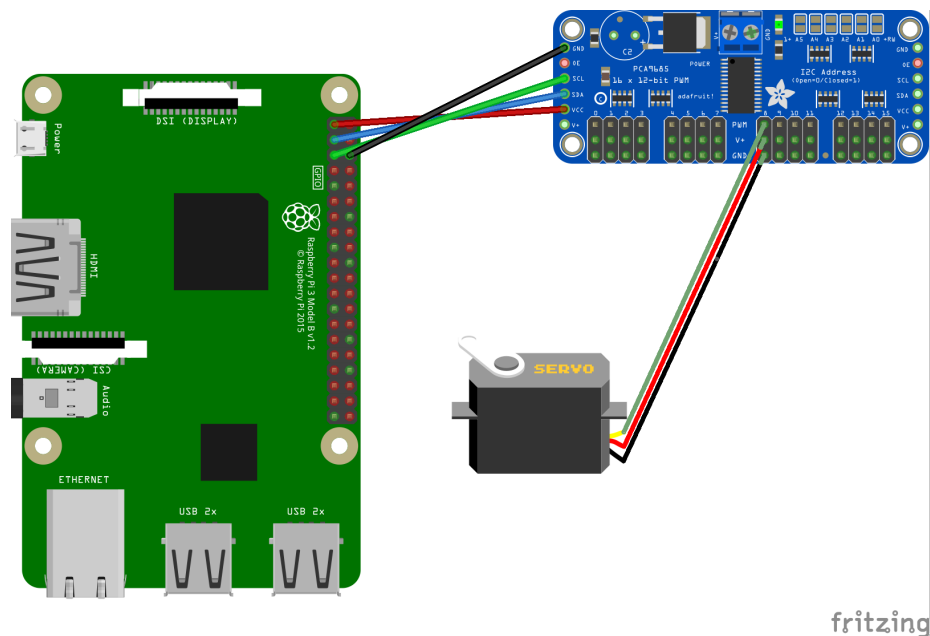


Figura 37 – Esquemático motores e RPi

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 a seguir.

```
@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'))
```

Figura 38 – Rotas de controle do motor

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.

```
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)
```

Figura 39 – Método *servo_position*

3.6.2 Configuração do Sensor PIR

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

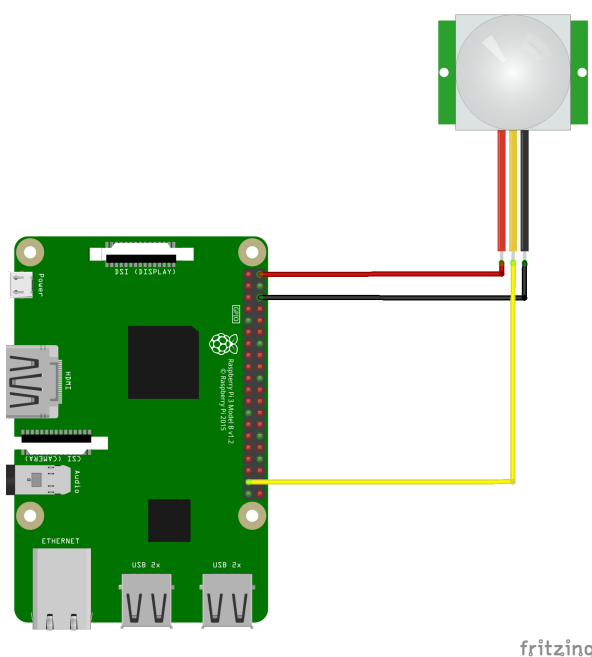


Figura 40 – Esquemático PIR

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* 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.

```
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")
```

Figura 41 – Método de configuração do sensor PIR

A função de *callback*, por sua vez, retorna o método *sendEmail* 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 (57). 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.

```
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")
```

Figura 42 – Método de envio do email

3.6.3 Configuração DHT11

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

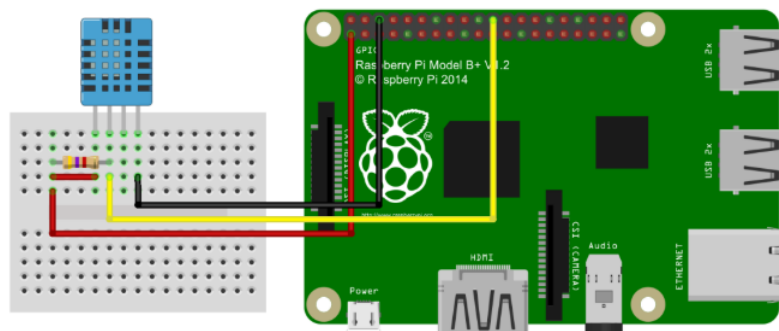


Figura 43 – Esquemático DHT11

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 e em seguida foi integrado ele ao código final. Para realizar a leitura do sensor DHT11, foi escrito o método *readDht11* dentro da classe *sensor*, conforme descrito a seguir:

```
def readDht11():
    umid, temp = Adafruit_DHT.read_retry(
        self.sensorTemperatureType,
        self.pinoTemperature
    )
    return umid, temp
```

Figura 44 – Método *readDht11*

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 é:

```

@app.route('/temperatura')
def temp():
    umidade, temperatura = sensor.readDht11();
    templateData = {'umid': umidade, 'temp': temperatura}

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

```

Figura 45 – Rota para receber os valores do sensor DHT11

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

3.6.4 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 da 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*".

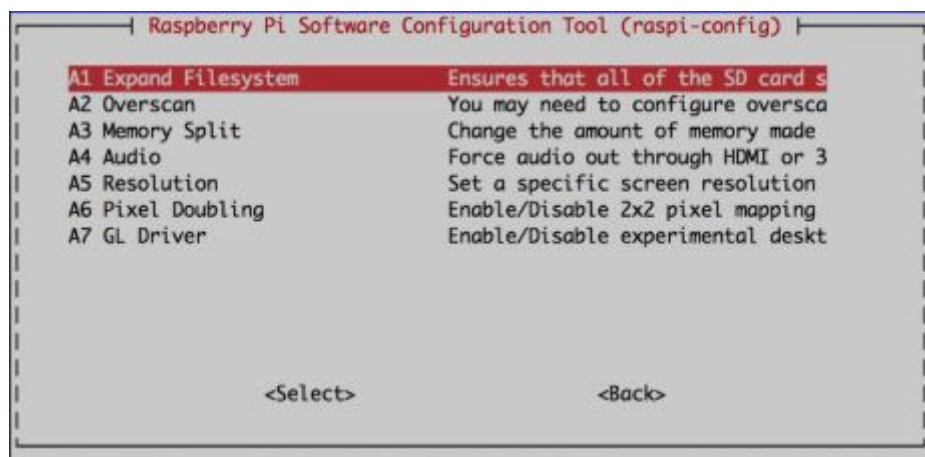


Figura 46 – Menu de Configuração RPi, Opções Avançadas

Em seguida "*Expand Filesystem*".

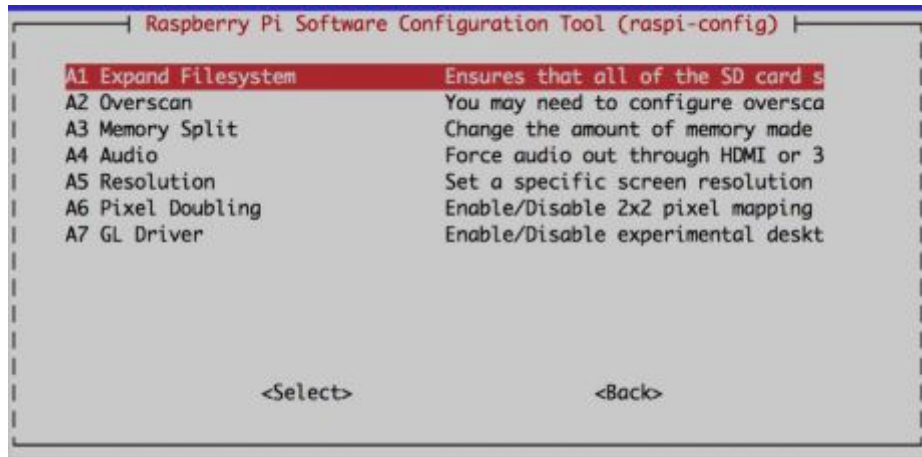


Figura 47 – Menu de Configuração RPi, Expandir Sistema

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 libcaneberra-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
```

3.6.4.1 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 a seguir:

```
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.

```
#!/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 = facecascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
```

Figura 48 – Método teste OpenCV

3.6.4.2 Implementação Geral

Para a implementação do código do OpenCV criou-se uma classe em um arquivo python. Onde que 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* está presente no Apêndice !!!!! e 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).

```
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
    )
```

Figura 49 – Método *runFaceDetector*

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 daí 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).

```
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
```

Figura 50 – Método *getImage*

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.

3.6.4.3 Implementação WEB

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

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

Figura 51 – Método *sendVideo*

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 remonta-la e envia-la para página WEB como uma imagem JPEG. Desta forma, devido ao loop a sequência de imagens assemelha-se a geração de um video em tempo real.

```
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')
```

Figura 52 – Método *generateVideo*, *server.py*

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).

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

Figura 53 – Endereço *cameraview*

3.6.5 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 (58):

- Análise 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 (59):

- 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 a baixo (60).

```
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 (61).

- 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 distancias (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* (62).

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

Figura 54 – Função *create_collection*

As imagens para treino são enviadas a partir da chamada da funcionalidade *index_faces* da API junto a definição dos seus parâmetros de entrada (62).

```
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'  
)
```

Figura 55 – Função *index_face*

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 minúsculas. A *CollectionId* recebe o nome da coleção criada para o desenvolvimento do projeto.

3.7 Comando de Voz

3.7.1 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* (23).

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 imagem.

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

Figura 56 – Configuração do arquivo config.txt

Para utilizar o speaker e o microfone é preciso indicar para a RPi quais são as portas correspondentes. O arquivo *asound.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.

```
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"
    }
}
```

Figura 57 – Configuração do arquivo *asound.conf*

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.

3.7.2 Desenvolvimento de Skills

3.8 Desenvolvimento do Banco de Dados

O banco de dados mostra-se fundamental pois será a partir dele que as instancias 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 ocorrerá 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 foi-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>”.

A seguir está presente o arquivo python que permite instanciar a tabela.

```
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)
```

Figura 58 – Banco de dados python (1)

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

```
import os
import sqlite3

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

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

Figura 59 – Banco de dados python (2)

4 Resultados e Discussões

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

4.1 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 disposto através de uma tela móvel como a presente no dispositivo.

4.1.1 Web

4.1.1.1 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 60 apresenta a página de entrada.

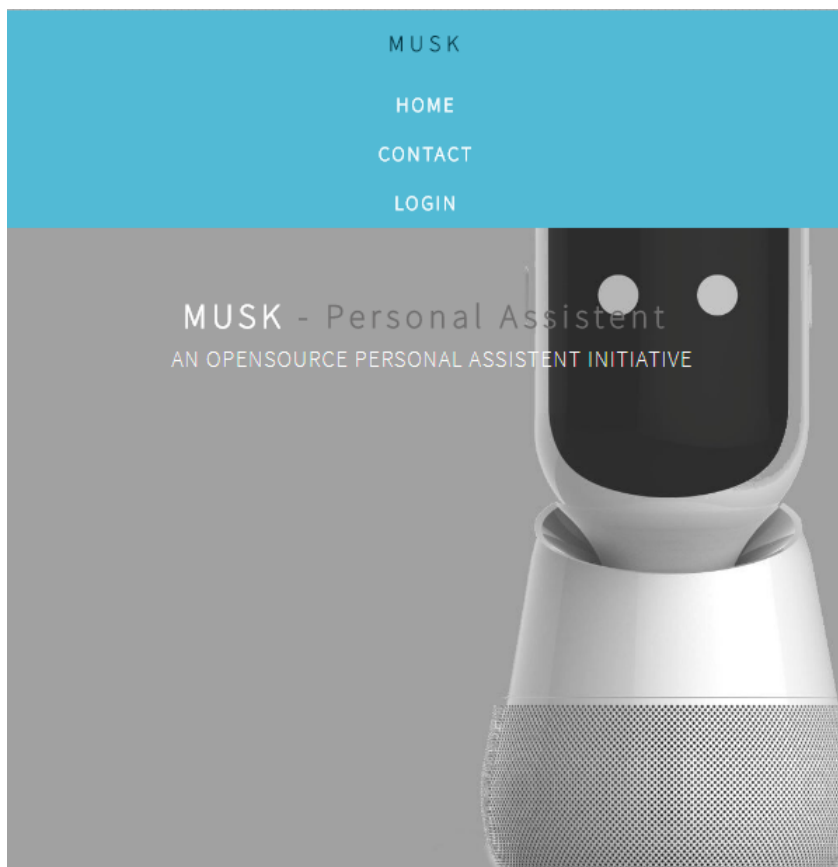


Figura 60 – Página de Entrada WEB

4.1.1.2 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 61 apresenta a home de entrada.



Figura 61 – Página de Aplicativos

4.1.1.3 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 62 apresenta um exemplo da interface desta função web.

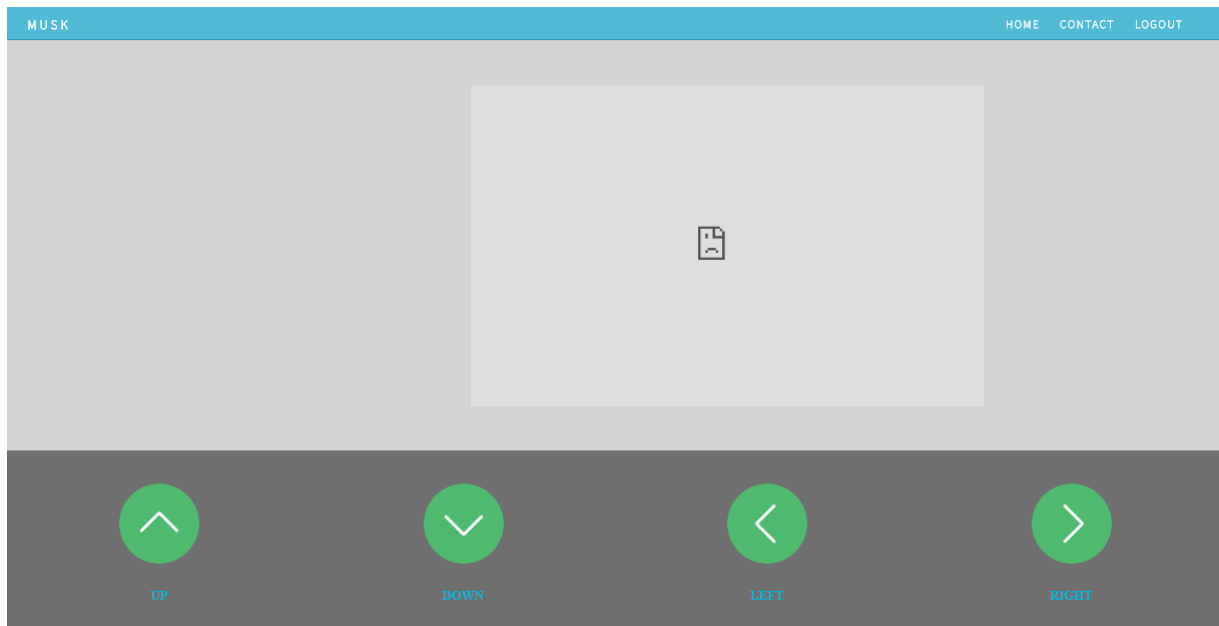


Figura 62 – Exemplo Interface Modo Vigia

4.1.1.4 Sensor de Temperatura e Umidade

A interface do sensor de umidade e temperatura está presente de forma simples em um exemplo na figura 63. Através dele é possível verificar localmente o ambiente e futuramente será possível até mesmo acionar determinados eletrodomésticos para controlar esta temperatura e torna-la ideal ao indicado pelo usuário.

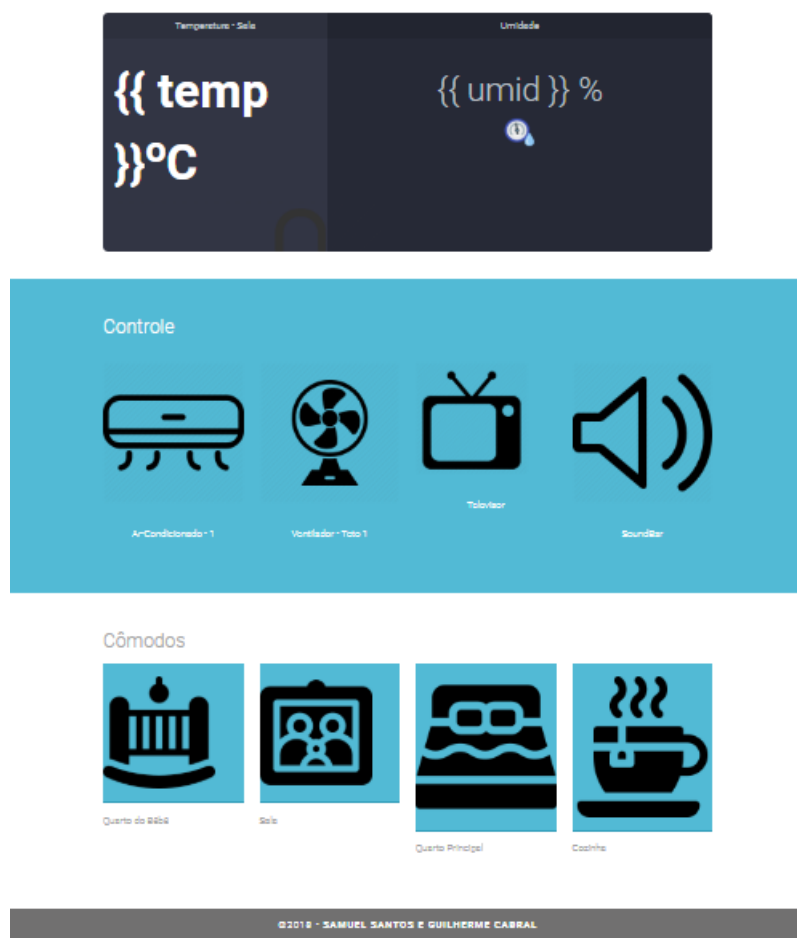





Figura 63 – Interface de Temperatura e Umidade

4.1.1.5 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.




Agende Suas Tarefas

Tarefas a **Finalizar**:

-  Affordable monthly premium packages
-  Choose your favourite doctor
-  Only use friendly environment

Agende Suas Tarefas

Tarefas a **Finalizadas**:

-  Affordable monthly premium packages
-  Choose your favourite doctor
-  Only use friendly environment

Adicione um Compromisso

Título	Last Name
<input type="text"/>	<input type="text"/>
Email	Data
<input type="text"/>	<input type="text"/>
SUBMIT	

Figura 64 – Interface Agenda

4.2 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 demonstrado a possibilidade de expansão do sistema com a integração de mais dispositivos. A seguir é apresentado os resultados obtidos para cada componente que compõe o sistema de segurança.

4.2.1 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 ótimos em relação ao tempo de resposta com um 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 a performance 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 o MUSK.

4.2.2 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 e uma sensibilidade alta. Em relação ao processo do email, obteve-se uma latência baixa e 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 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.

4.2.3 Detecção e 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 da etapa de detecção e reconhecimento.

4.2.3.1 Testes Detecção de Faces

4.2.3.2 Testes AWS Rekognition

Para verificar a confiabilidade e latência da API da amazon, foi desenvolvido scripts automatizados 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 era realizar um número considerável de testes (processo 1) e verificar se existe alguma perda de performance em imagens tiradas 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 a seguir mostra os resultados obtidos para um total de 800 amostras.

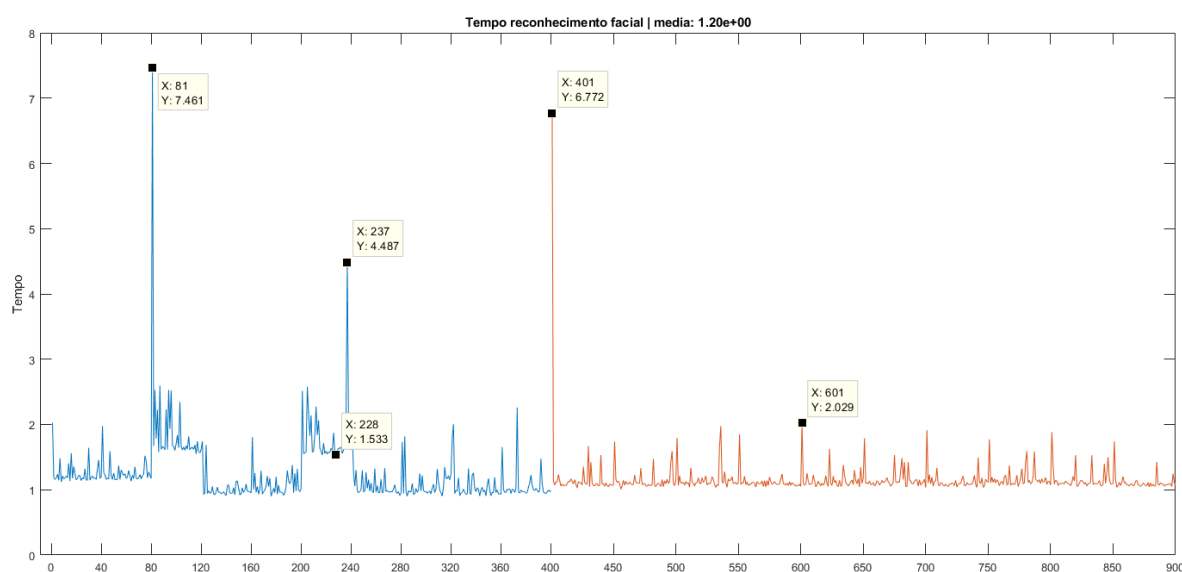


Figura 65 – Teste da Variação do Tamanho da Imagem Enviada

As fotos de tamanho original estão alocadas na primeira banda de amostras (0 - 400). Estas obtiveram uma instabilidade em relação o 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 satisfatórios e 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 a seguir.

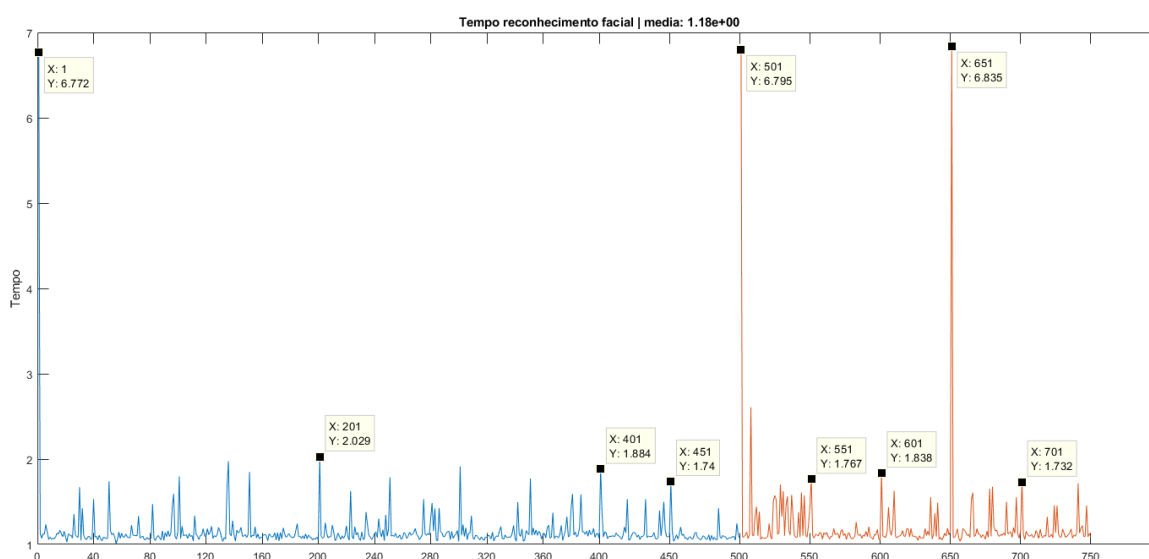


Figura 66 – Picos de tempo no início do teste

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 pessoa 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 abaixo.

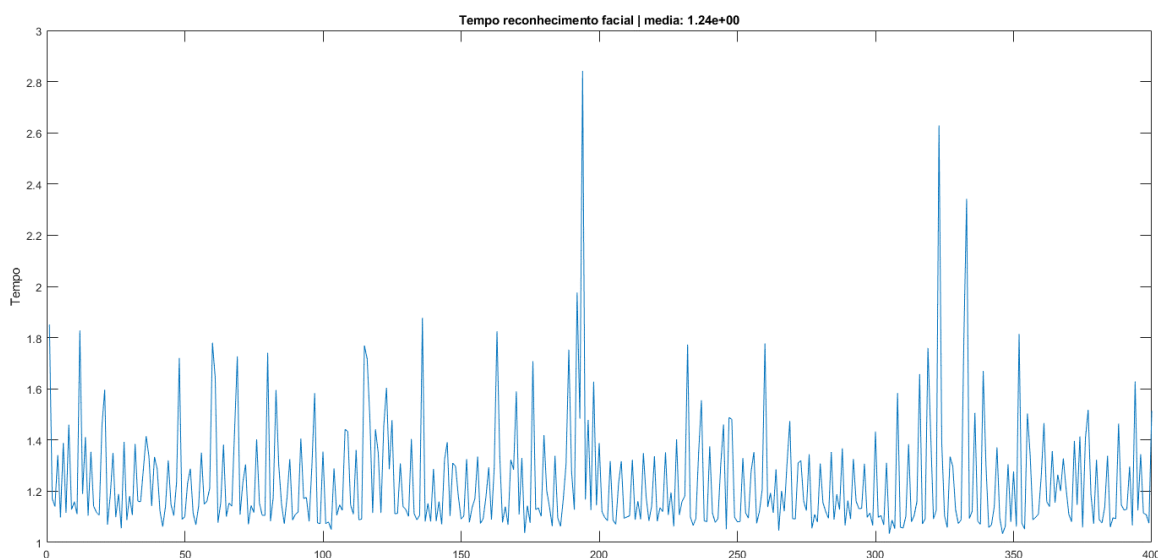


Figura 67 – Teste com variação da imagem de entrada

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 a seguir.

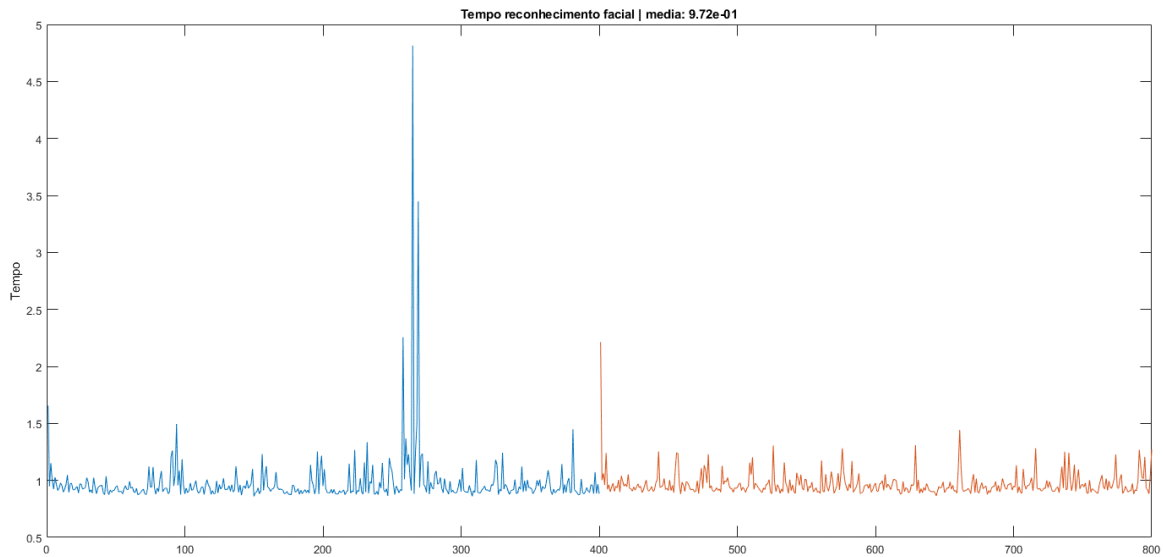


Figura 68 – Teste em tempo real

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%.

4.3 Controle de Voz

4.4 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 a seguir. O período de avaliação do funcionamento do assistente foi de 38 minutos, com início às 17 horas e 15 minutos.



Figura 69 – Teste de temperatura

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 70°C. 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 - 45°C, observa-se que a raspberry atingiu alguns picos críticos de aquecimento.

4.5 Aprimoramentos Futuros

Devido ao tempo limitado do projeto e devido ao seu objetivo principal que é de servir como base para desenvolvimentos futuros na área, o assistente pessoal – MUSK – ,ainda tem muito a evoluir. Alguns dos elementos interessantes para uma abordagem futura são o reconhecimento facial, o reconhecimento e comando por voz e a capacidade de responder por voz os comandos recebidos.

Com o intuito de tornar estes aprimoramentos mais tangíveis foi-se realizada uma pesquisa sobre cada tema. Nesta pesquisa observou-se algumas das tecnologias que podem auxiliar na construção destas funcionalidades.

O reconhecimento facial pode ser realizado através da biblioteca de visão computacional OpenCV, na qual pode-se aplicar também refinamentos e aprendizado de máquina para elevar a eficiência e velocidade de processamento. Já o comando de voz e a resposta por voz pode ser realizada através de algumas API's de “speech recognition” fornecidas pela google que são compatíveis com o Python.

Outro aprimoramento futuro também é a melhoria no design e na experiência de usuário que se dá a partir de um melhor entendimento quanto ao funcionamento do Kivy, de JavaScript e CSS.

5 Conclusão

A partir deste trabalho foi possível desenvolver e aprimorar os conhecimentos de diversas linguagens de programação e marcação, assim como python, SQL, HTML, JavaScript e CSS. Além disso, pode-se observar a integração entre elementos digitais e Web com componentes eletromecânicos (servo motor) e sensores.

O desenvolvimento do assistente pessoal MUSK permitiu que os conhecimentos obtidos ao longo dos anos na teoria pudessem ser combinados e aplicados de forma prática. É importante ressaltar que os objetivos iniciais presentes no escopo do projeto não foram completamente concluídos, devido aos recursos e ao período limitado para sua construção.

Conforme previsto, a interface Web foi finalizada, contando com as páginas: inicial, aplicativos, modo vigia, temperatura e umidade e agenda. Os sensores foram programados e testados em bancada, possibilitando a integração deles junto ao corpo principal do projeto.

Apesar das horas extras e do esforço apresentado pela equipe, não foi possível implementar o reconhecimento facial e de voz. No entanto, uma vez que o trabalho visa servir de base para o desenvolvimento futuro de um assistente pessoal mais completo, pode-se afirmar que o trabalho concluiu seus objetivos.

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 10.
- [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 10.
- [3] “Apple future computer knowledge navigator.” <https://www.businessinsider.com/apple-future-computer-knowledge-navigator>. Accessed: 2019-10-15. Citado na página 10.
- [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 10.
- [5] “Assistentes pessoais digitais.” <https://oglobo.globo.com/economia/assistentes-pessoais-digitais-1-19383632>. Accessed: 2019-10-15. Citado na página 10.
- [6] “Internet das coisas: Um desenho do futuro.” <https://www.proof.com.br/blog/internet-das-coisas/>. Accessed: 2019-10-15. Citado na página 10.
- [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 10.
- [8] W. STALLINGS, *Arquitetura e organização de computadores*. Pearson Prentice-Hall, 10^a ed., 2017. Citado na página 13.
- [9] M. Schmidt, *Raspberry Pi: A Quick-Start Guide*. Pragmatic Bookshelf, 2014. Citado 2 vezes nas páginas 14 e 24.
- [10] “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 14.
- [11] “Processador arm.” http://www.dca.fee.unicamp.br/~lbocato/topico_3.1_processador_ARM.pdf. Accessed: 2018-06-20. Citado na página 14.

- [12] “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 15.
- [13] “3.5inch rpi display.” http://www.lcdwiki.com/3.5inch_RPi_Display. Accessed: 2019-10-14. Citado na página 15.
- [14] D. Thomazini, *Sensores industriais: fundamentos e aplicações*. Saraiva Educação SA, 2018. Citado na página 17.
- [15] “Sinal analógico x sinal digital.” <https://www.embarcados.com.br/sinal-analogico-x-sinal-digital/>. Accessed: 2019-10-14. Citado na página 17.
- [16] P. HC-SR501, “Motion detector datasheet,” *Product Description*. Citado na página 18.
- [17] W. W. Gay, “Dht11 sensor,” in *Experimenting with Raspberry Pi*, pp. 1–13, Springer, 2014. Citado na página 19.
- [18] “Manual webcam goldship 3817.” Manufacturer: GoldShip, Inc, LeaderSheep. Citado na página 20.
- [19] “Microfone universal mini usb 2.0.” Manufacturer: Mini, Inc. Citado na página 21.
- [20] “O que é servomotor? controlando um servo com arduino.” <https://portal.vidadesilicio.com.br/o-que-e-servomotor/>. Accessed: 2019-10-14. Citado na página 22.
- [21] A. S. Tanenbaum and H. Bos, *Modern operating systems*. Pearson, 2015. Citado 2 vezes nas páginas 23 e 24.
- [22] “About raspbian.” <https://www.raspbian.org/RaspbianAbout>. Accessed: 2019-10-14. Citado 2 vezes nas páginas 24 e 25.
- [23] “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 25 e 69.
- [24] “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 25.
- [25] “Como funcionam as aplicações web.” <https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888>. Accessed: 2019-10-14. Citado na página 26.

- [26] “Métodos de requisição http.” <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Accessed: 2019-10-14. Citado na página 26.
- [27] “What does “micro” mean?.” <https://flask-doc.readthedocs.io/en/latest/foreword.html>. Accessed: 2019-10-14. Citado na página 26.
- [28] “Python flask introdução.” <http://devfuria.com.br/python/flask/>. Accessed: 2019-10-14. Citado na página 26.
- [29] “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 26.
- [30] “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 26.
- [31] “Html - standards.” <https://www.w3.org/standards/>. Accessed: 2019-06-20. Citado na página 27.
- [32] A. WESLEY, *A history of HTML*. 1998. Citado na página 27.
- [33] “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 2 vezes nas páginas 27 e 28.
- [34] H. F. KORTH and A. SILBERCHATZ, “Sistemas de banco de dados. rev.” Citado na página 29.
- [35] C. J. Date, *Introdução a sistemas de bancos de dados*. Elsevier Brasil, 2004. Citado na página 29.
- [36] J. Laudon, Kenneth; Laudon, *Sistemas de Informação Gerenciais*. Pearson Brasil, 2011. Citado na página 30.
- [37] “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 30.
- [38] “Sql.” <https://www.sqlite.org/features.html>. Accessed: 2018-05-06. Citado na página 31.
- [39] “O que é visão computacional?.” <http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>. Accessed: 2019-10-14. Citado na página 31.

- [40] “O que é visão computacional?” <https://blogbrasil.comstor.com/o-que-e-visao-computacional>. Accessed: 2019-10-14. Citado na página 31.
- [41] “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 32.
- [42] “Opencv about.” <https://opencv.org/about/>. Accessed: 2019-10-14. Citado na página 32.
- [43] “Opencv documentation.” <https://docs.opencv.org/4.1.1/d1/dfb/intro.html>. Accessed: 2019-10-14. Citado na página 32.
- [44] “O que é a computação em nuvem?” <https://aws.amazon.com/pt/what-is-cloud-computing/>. Accessed: 2019-10-14. Citado na página 33.
- [45] “Amazon rekognition.” <https://aws.amazon.com/pt/rekognition/>. Accessed: 2019-10-14. Citado na página 33.
- [46] “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 34.
- [47] “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>. Accessed: 2019-10-14. Citado na página 35.
- [48] “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 37.
- [49] “Dynamic Host Configuration Protocol (DHCP) kernel description.” <https://www.raspberrypi.org/learning/networking-lessons/lesson-3/plan/>. Accessed: 2018-06-22. Citado na página 37.
- [50] “MAC address spoofing kernel description.” https://wiki.archlinux.org/index.php/MAC_address_spoofing. Accessed: 2018-06-22. Citado na página 38.
- [51] “SSH (SECURE SHELL) kernel description.” <https://www.raspberrypi.org/documentation/remote-access/ssh/>. Accessed: 2018-06-22. Citado na página 38.

- [52] “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 39.
- [53] “Sites responsivos com muita rapidez..” https://www.adobe.com/br/products/dreamweaver.html?gclid=Cj0KCQjw3JXtBRC8ARIsAEBHg4k4GxQiSHlz0I7fXA_YecH2e3Up9_ExjntHECwgIrRZDElNihMCC0YaAuUZEALw_wcB&sdid=KQPQE&mv=search&ef_id=Cj0KCQjw3JXtBRC8ARIsAEBHg4k4GxQiSHlz0I7fXA_YecH2e3Up9_ExjntHECwgIrRZDElNihMCC0YaAuUZEALw_wcB:G:s&s_kwid=AL!3085!3!301784449881!e!!g!!dreamweaver. Accessed: 2019-10-14. Citado na página 40.
- [54] “Raspberry Temp. Server kernel description.” <http://raspberrypiwebserver.com/cgisc scripting/rpi-temperature-logger/building-a-web-user-interface-for-the-temperature-monitor.html>. Accessed: 2018-06-22. Citado na página 42.
- [55] “Raspberry Forms Server kernel description.” <http://raspberrypiwebserver.com/cgisc scripting/web-forms-with-python.html>. Accessed: 2018-06-22. Citado na página 43.
- [56] “XML HttpRequest kernel description.” https://www.w3schools.com/xml/xml_http.asp. Accessed: 2018-06-22. Citado na página 43.
- [57] “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 57.
- [58] “Definição de preço do amazon rekognition.” <https://aws.amazon.com/pt/rekognition/pricing/?nc=sn&loc=4>. Accessed: 2019-10-14. Citado na página 66.
- [59] “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 67.
- [60] “Aws sdk para python (boto3).” <https://aws.amazon.com/pt/sdk-for-python/>. Accessed: 2019-10-14. Citado na página 67.
- [61] “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 67.

-
- [62] “Available services.” <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html>. Accessed: 2019-10-14. Citado na página 68.

Apêndices

APÊNDICE A – Código Página Principal

HTML

```
1 <!doctype html>
  <html lang="pt-BR">
3 <head>
  <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
7 <title>MUSK</title>
  <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.-->
  <script>var __adobewebfontsappname__="dreamweaver"</script>
11 <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="
    text/javascript"></script>

13 <style>

15   .hero{
    background-image:url(static/images/logo1.jpg);
17    background-size: cover;          /* For flexibility */
    height: 290px

19    }

21   .banner{
    height: 160px;
23    margin-top: 70px;
    }
25 </style>

27 </head>
  <body>

29   <!-- Main Container -->
31 <div class="container">
  <!-- Navigation -->
33 <header> <a href="">
    <h4 class="logo">MUSK</h4>
35 </a>
```

```

37     <nav>
    <ul>
        <li><a href="{{url_for('index')}}">HOME</a></li>
39        <li><a href="{{url_for('index')}}">CONTACT</a></li>
    <li><a href="{{url_for('index2')}}">LOGIN</a></li>
41    </ul>
    </nav>
43 </header>
    <!-- Hero Section -->
45 <section class="hero" id="hero">

47     <h2 class="hero_header">MUSK <span class="light">- Personal Assistant </span></h2>
    <p class="tagline"> AN opensource personal assistant INITIATIVE</p>
49 </section>
    <!-- About Section -->
51 <section class="about" id="about">
53     <h2 class="hidden">About</h2>
    <p class="text_column">O Musk um projeto de assistente pessoal de
        c digo 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>Ent o esse o Musk um projeto para todos
        , espero que goste. </p>
    </section>
57     <!-- Parallax Section -->
59 <section class="banner">
    <h2 class="parallax">O que o MUSK pode fazer por voc ?</h2>
61 </section>
    <!-- More Info Section -->
63 <footer>
    <article class="footer_column">
65     <h3>MODO VIGIA</h3>
    
67     <p>Uma das grandes qualidades do MUSK a de ser os seus olhos e
        ouvidos quando voc
        n o est em casa, mas quer saber como as coisas andam por l </p>
69 </article>
    <article class="footer_column">
71     <h3>TEMPERATURA</h3>
    
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 B – Código HTML Página Aplicativos

```
1  <!doctype html>
3  <html lang="pt-br">
   <head>
5     <meta charset="utf-8">
     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
     <link href="{{ url_for('static', filename='css/simpleGridTemplate.css') }}"
         rel="stylesheet" type="text/css">
9
   <title>MUSK</title>
11  <link href="{{ url_for('static', filename='stylesheets/css/
    singlePageTemplate.css') }}" rel="stylesheet" type="text/css">
     <script>var __adobewebfontsappname__="dreamweaver"</script>
13  <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="
    text/javascript"></script>

15  <style>
     .title_{
17     grid-template: up;
     }
19
   </style>
21
   </head>
23  <body>
     <!-- Main Container -->
25  <div class="container">
     <!-- Navigation -->
27
     <header id="logo">
29
         <a class="logo"></a>
31
     <nav>
         <ul>
33
         <li id = "title_musk"><a href="" >MUSK</a></li>
         <li><a href="{{ url_for('index2') }}">HOME</a></li>
```

```

37     <li> <a href="{{url_for('index')}}">CONTACT</a></li>
    <li><a href="{{url_for('index')}}">LOGOUT</a></li>
39 </ul>
    </nav>
41 </header>
    <!-- Hero Section -->
43 <section class="intro">
    <div class="column">
45     <h3>JOHN DOE</h3>
     </div>
47 <div class="column">
    <p>Olá, essa é a sua página principal e aqui estão disponíveis
        todos os aplicativos desenvolvidos com o foco de facilitar o seu
        dia a dia. Futuramente se desenvolverá uma loja que permitirá
        que você baixe e se utilize apenas os aplicativos que você
        deseja, mas até lá aproveite todos os nossos serviços </p>
49 <p>Lembre-se sempre que estamos de ouvidos abertos a melhorias,
        assim caso surja uma nova ideia ou sugestão basta nos contactar.
        Estamos trabalhando cada vez mais para deixar esse projeto o mais
        amigável possível. </p>
    </div>
51 </section>
    <!-- Stats Gallery Section -->
53 <div class="gallery">
    <div class="thumbnail"> <a href="{{url_for('camera_display')}}"></a>
    <h4>MODO VIGIA</h4>
    <p class="tag">CAMERA</p>
    <p>Deixe que nós sejamos seus olhos e ouvidos<br>.</p>
    </div>
59 <div class="thumbnail"> <a href="{{url_for('temp')}}"></a>
    <h4>TEMPERATURA</h4>
    <p class="tag">COMO ESTÁ A CASA</p>
    <p class="">Veja como deixar o seu ambiente mais confortável</p>
63 </div>
    <div class="thumbnail"> <a href="{{url_for('agenda')}}"></a>
65 <h4>AGENDA</h4>
    <p class="tag">O QUE TEMOS PARA HOJE ?</p>
    <p>Veja aqui quais são as suas próximas tarefas <br>.</p>
67 </div>
69 </div>
71 <!-- Copyrights Section -->

```

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

APÊNDICE C – Controle Motores e Camera

HTML

```
1 <!doctype html>
  <html>
3 <head>
  <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
7 <title>MUSK</title>
  <link href="{{ url_for('static', filename='css/multiColumnTemplate.css') }}"
    " rel="stylesheet" type="text/css">
9
  <script>var __adobewebfontsappname__="dreamweaver"</script>
11 <script src="http://use.edgefonts.net/source-sans-pro:n2:default.js" type="
    text/javascript"></script>
  <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media
    queries -->
13 <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
  <!--[if lt IE 9]>
15     <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js
        "></script>
        <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></
          script>
17     <![endif]-->

19   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js
        "></script>
    <script language="javascript" type="text/javascript">
21     function Button_onclick(direction) {
        $.ajax({ url: '/' + direction })
23     }
    </script>
25
  </head>
27 <body style="background-color: transparent;">
  <div class="container">
29 <header> <a href="">
    <h4 class="logo">MUSK</h4>
31   </a>
  <nav>
33   <ul>
```

```

35     <li id = "title_musk"><a href="{{url_for( 'index2' )}}" >MUSK</a></li>
        <li><a href="{{url_for( 'index2' )}}">HOME</a></li>
        <li> <a href="{{url_for( 'contact' )}}">CONTACT</a></li>
37     <li><a href="{{url_for( 'index' )}}">LOGOUT</a></li>

39     </ul>
    </nav>
41 </header>
<section class="" style="background: linear-gradient(to right, lightgray ,
    lightgray)">
43 <p align="right" style="margin: inherit; margin-right: 300px;"><iframe
    scrolling="no" frameborder="0" src="http://192.168.0.46:8081"></iframe
    ></p>
</section>
45 <div class="row" style="margin-bottom: 400px;">
    <div class="columns">
47     <p class="thumbnail_align"> <button style="background-color:
        transparent; border: none;">
        
49     </button>
        </p>
51     <h4>UP</h4>
</div>
53     <div class="columns">
        <p class="thumbnail_align"> <button style="background-color:
            transparent; border: none;">
55         
            </button> </p>
57         <h4>DOWN</h4>
</div>
59     <div class="columns">
        <p class="thumbnail_align"> <button style="background-color:
            transparent; border: none;">
61         
            </button> </p>
63         <h4>LEFT</h4>
</div>
65     <div class="columns">
        <p class="thumbnail_align"> <button style="background-color:
            transparent; border: none;">
67         
            </button> </p>
69         <h4>RIGHT</h4>

```



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

APÊNDICE D – 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/font-awesome.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>
```

```

35     <nav>
        <ul>
            <li><a style="font-family: 'source-sans-pro';" href="{{url_for('
37             index2')}}">HOME</a></li>
            <li><a style="font-family: 'source-sans-pro';" href="{{url_for('
                contact')}}">CONTACT</a></li>
            <li><a style="font-family: 'source-sans-pro';" href="{{url_for('
                index')}}">LOGOUT</a></li>
39         </ul>
    </nav>
41 </header>

43 <!-- .site-header -->

45
47 <div class="forecast-table" style="background-color: white;">
    <div class="container" style="margin-top: 190px;">

49        <div class="forecast-container">
            <div class="today forecast">
51                <div class="forecast-header">
                    <div class="day" style="float: none;">Temperatura - Sala</
                        div>

53                </div> <!-- .forecast-header -->
                    <div class="forecast-content">
85                        <div class="degree">
                            <div class="num">{{ temp }}<sup>o</sup>C</div>
                            <div class="forecast-icon">
59                                
                            </div>
61                        </div>
                        </div>
63                    </div>
                <div class="forecast">
55                    <div class="forecast-header">
                        <div class="day"> Umidade</div>
67                    </div> <!-- .forecast-header -->
                        <div class="forecast-content">
69                            <div class="num" style="font-size: 62px;">{{ umid }} %</div>
                                >
                            <div class="forecast-icon">
71                                
                            </div>
73                        </div>
                    </div>

```

```

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                             img src="static/images/ar-cond-1.png" alt="" width="50
89                             " height="5" ></figure>
90                         <h3 align="center" class="location">Ar-Condicionado - 1</
91                         h3>
92                     </div>
93                 </div>
94                 <div class="col-md-3 col-sm-6">
95                     <div class="live-camera">
96                         <figure class="live-camera-cover" style="border: none;"><
97                             img src="static/images/vent-1.png" alt=""></figure>
98                         <h3 align="center" class="location">Ventilador - Teto 1</
99                         h3>
100                     </div>
101                 </div>
102                 <div class="col-md-3 col-sm-6">
103                     <div class="live-camera">
104                         <figure class="live-camera-cover" style="border: none; "
105                             ></figure>
107                         <h3 align="center" class="location">Televisor</h3>
108                     </div>
109                 </div>
110                 <div class="col-md-3 col-sm-6">
111                     <div class="live-camera">
112                         <figure class="live-camera-cover" style="border: none;"><
113                             img src="static/images/sound-1.png" alt=""></figure>
114                         <h3 align="center" class="location">SoundBar</h3>
115                     </div>
116                 </div>
117             </div>
118         </div>
119     </div>

```

```

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>
126           <h3 class="location" style="color: #A3A3A3;">Quarto do
127             B b </h3>
128         </div>
129       </div>
130       <div class="col-md-3 col-sm-6">
131         <div class="live-camera">
132           <figure class="live-camera-cover"></figure>
135           <h3 class="location" style="color: #A3A3A3;">Sala</h3>
136         </div>
137       </div>
138       <div class="col-md-3 col-sm-6">
139         <div class="live-camera">
140           <figure class="live-camera-cover"></figure>
142           <h3 class="location" style="color: #A3A3A3;">Quarto
143             Principal</h3>
144         </div>
145       </div>
146       <div class="col-md-3 col-sm-6">
147         <div class="live-camera">
148           <figure class="live-camera-cover"></figure>
150           <h3 class="location" style="color: #A3A3A3;">Cozinha</h3>
151         </div>
152       </div>
153     </div>
154   </div>
155 </div>
156 </main> <!-- .main-content -->
157
158 <div class="copyright"> &copy;2018 - <strong>Samuel Santos e Guilherme
159   Cabral</strong> </div>

```

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

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

157 </body>

159 </html>
```

APÊNDICE E – Programa Principal em Python

```
1 # -*- coding: utf-8 -*-
  from __future__ import division
3 from flask import Flask, render_template, Response, redirect, url_for,
    request, abort
  from flask_sqlalchemy import SQLAlchemy
5 from datetime import datetime

7 import threading
  import os
9 import commands
  import time
11 import Adafruit_PCA9685
  import Adafruit_DHT
13 import RPi.GPIO as GPIO
  import smtplib
15 import schedule
  import time

17
  import kivy
19 from kivy.app import App
  from kivy.lang import Builder
21 from kivy.ui.screenmanager import ScreenManager, Screen, NoTransition
  from kivy.ui.button import Button
23 from kivy.clock import Clock
  from kivy.ui.widget import Widget
25 from kivy.ui.floatlayout import FloatLayout
  from kivy.properties import ObjectProperty
27 from kivy.ui.boxlayout import BoxLayout
  from kivy.ui.scrollview import ScrollView
29 import sqlite3

31 # ----- CONFIG KIVY -----
  kivy.require('1.9.0')
33
  conn = sqlite3.connect('todo.db') # cria se necessario a base da dados
35 c = conn.cursor() # gera o cursor da db

37 # You can create your kv code in the Python file
  Builder.load_string("""
```

```
39 | <ScreenOne@Button>:
41 |
43 |     background_color: 0,0,0,1
45 |
47 |     on_press:
49 |         root.manager.current = 'screen_two'
51 |
53 |     BoxLayout:
55 |         text: "MUSK"
57 |
59 |         Image:
61 |             source: 'git.gif'
63 |             size_hint_y: None
65 |             height: 300
67 |             size: 400, 400
69 |             allow_stretch: True
71 |
73 | <ScreenTwo@Button>:
75 |
77 |     background_color: 0,0,0,1
79 |
81 |     on_press:
83 |         root.manager.current = 'screen_three'
85 |
87 |     Button:
89 |         background_color: 0,0,0,1
91 |         text: root.registrador_umid_e_temp
93 |         size: 150,50
95 |         font_size: 36
97 |         pos_hint: {"center_x": .5, "center_y": .65}
99 |
101 | """
103 |
105 | app = Flask(__name__)
107 |
109 | # ——— CONFIG AGENDA ———
111 | app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///home/pi/web-server/todo
```



```

    .db'
db = SQLAlchemy(app)
87
class Todo(db.Model):
89     id = db.Column(db.Integer, primary_key=True)
        title = db.Column(db.String(200))
91     start = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
        complete = db.Column(db.Boolean)
93
95
# ----- CONFIG SERVO MOTOR -----
97 pwm = Adafruit_PCA9685.PCA9685()
    servo_min = 150 # Min pulse length out of 4096
99    servo_max = 600 # Max pulse length out of 4096
    pwm.set_pwm_freq(60)
101 Memory = [0,0]

103 def step(x):
        y = 375 + 25*x
105         if y>=600:
                y = 600
107         if y<=150:
                y = 150
109         return y

111 def servo_position(Motor, pos):
        #Motor = abs(Motor)
113         if pos == 1:
                Memory[Motor]+=1
115         if pos == -1:
                Memory[Motor]-=1
117
        pulse = step(Memory[Motor])
119        Motor = Motor + 2
        pwm.set_pwm(Motor, 0, pulse)
121
123
# ----- INICIA CAMERA -----
125 os.system('sudo service motion start')
127
129 # ----- CONFIG DHT11 -----
    sensor = Adafruit_DHT.DHT11
131 GPIO.setmode(GPIO.BOARD)

```

```
pino_sensor = 25
133
135
137 # ----- CONFIG PIR -----
PIR_PIN = 26
GPIO.setup(PIR_PIN, GPIO.IN)
139
141
143 # ----- CONFIG ENVIAR EMAIL -----
# Credenciais
remetente = 'projetosd373@gmail.com'
145 senha = 'projetosel373'
147
# Informa es da mensagem
destinatario = 'gui.cabral201@gmail.com'
149 assunto = 'PROJETO SD'
texto = 'Presen a Detectada!'
151
# Preparando a mensagem
153 msg = '\r\n'.join([
    'From: %s' % remetente,
155 'To: %s' % destinatario,
    'Subject: %s' % assunto,
157 ', ',
    '%s' % texto
159 ])
161
163 # ----- REQUISICOES -----
@app.route('/')
165 def index():
    """Video streaming home page."""
167     return render_template('home_entry.html')
169
def start_app():
171     print("Starting Flask app...")
    app.run(host='0.0.0.0', threaded=True)
173
175 @app.route('/inside')
def index2():
177     return render_template('home_inside.html')
```

```
179 @app.route('/camera')
181 def camera_display():
182     """Video streaming home page."""
183     return render_template('camera.html')
185
186 @app.route('/panleft')
187 def move2():
188     servo_position(1,-1)
189     return redirect(url_for('camera_display'))
191
192 @app.route('/panright')
193 def move3():
194     servo_position(1,1)
195     return redirect(url_for('camera_display'))
197
198 @app.route('/tiltup')
199 def move4():
200     servo_position(0,1)
201     return redirect(url_for('camera_display'))
203
204 @app.route('/tiltdown')
205 def move5():
206     servo_position(0,-1)
207     return redirect(url_for('camera_display'))
209
210 @app.route('/contact_pag')
211 def contact():
212     return redirect(url_for('index'))
213
214 @app.route('/temperatura')
215 def temp():
216
217     # Efetua a leitura do sensor
218     umidade, temperatura = Adafruit_DHT.read_retry(sensor, pino_sensor);
219
220     templateData = {
221         'umid': umidade,
222         'temp': temperatura
223     }
225
```

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

229 @app.route('/agenda_pag')
231 def agenda():
233     incomplete = Todo.query.filter_by(complete=False).all()
235     complete = Todo.query.filter_by(complete=True).all()

237     return render_template('Agenda.html', incomplete=incomplete, complete=
        complete)

239 @app.route('/add', methods=['POST'])
241 def add():
243     todo = Todo(title=request.form['todoitem'], start=datetime.strptime(
        request.form['date'], '%Y-%m-%d'), complete=False)
245     db.session.add(todo)
247     db.session.commit()

249     return redirect(url_for('agenda'))

251 @app.route('/complete/<id>')
253 def complete(id):
255     todo = Todo.query.filter_by(id=int(id)).first()
257     todo.complete = True
259     db.session.commit()

261     return redirect(url_for('agenda'))

263 # ——— KIVY ———
265 # gera se necessario a tabela do db
267 def create_table():
269     c.execute("CREATE TABLE IF NOT EXISTS Todo(id INTEGER, title TEXT,
        start DATETIME, complete NUMERIC)")

271 #Teste de entrada de dados na tabela
273 def data_entry():
275     c.execute("INSERT INTO Todo VALUES(145, 'ujadssakd', '2016-01-11
        13:53:39', 1)")
277     conn.commit()
279     c.close()
281     conn.close()
```

```
269
271 #L   na tabela determinada coluna – nesse caso a title –
def read_from_db():
273
    c.execute('SELECT title FROM Todo WHERE complete = 1 ')
275     data = c.fetchall()
277     return data
279 #transforma o read_from_db em string e elimina os caracteres SQL
def read_string_db():
281
283     data = read_from_db()
    len_data = len(data)
285     data_str = data
287
    for i in range(len_data):
        var1 = str(data[i])
289         len_data_str = len(var1)
        var1 = var1[2:(len_data_str-4)] # recorta string https://pt.
            stackoverflow.com/questions/272561/como-remover-caracteres-de-
            posi%C3%A7%C3%A3o-espec%C3%ADfica-de-uma-string
291         data_str[i] = var1
293     return data_str
295 class ScreenOne(Screen, Button):
297
    def __init__(self, **kwargs):
        super(ScreenOne, self).__init__(**kwargs)
299         # Clock.schedule_interval(self.callback, 3) #Chama a callback
            depois de um tempo
301
    def callback(self, dt):
        # print('In Callback') # Test – The timer is actually calling this
            function
303         screen_manager.transition = NoTransition()
        screen_manager.current = 'screen_two'
305
307 class ScreenTwo(Screen, Button):
    umid = 9
309     umid = "Umidade: " + str(umid) + " g/Kg \n"
311
    temp = 10
```

```

temp = "Temperatura: " + str(temp) + " C \n"
313
now = datetime.now()
315 date = "\n" + "\n" + "\n" + "Hoje: " + str(now.day) + "/" + str(now.
    month) + "/" + str(now.year) + " \n"
time = "Horas: " + str(now.hour) + ":" + str(now.minute) + ":" + str(
    now.second)
317
registrador_umid_e_temp = ObjectProperty(None)
319 registrador_umid_e_temp = temp + umid + date + time

321 def __init__(self, **kwargs):
    super(ScreenTwo, self).__init__(**kwargs)
323     # Clock.schedule_interval(self.callback, 4)

325 def callback(self, dt):
    # print('In Callback') # Test - The timer is actually calling this
    function
327     screen_manager.transition = NoTransition()
    screen_manager.current = 'screen_one'
329

331 class ScreenThree(Screen):
    def __init__(self, **kwargs):
333         super(ScreenThree, self).__init__(**kwargs)
        layout = BoxLayout(orientation="vertical", size_hint_y=None)
335         layout.bind(minimum_height=layout.setter('height'))

337         title = read_string_db()
        len_title = len(title)
339

        ''' Explica o ( e Consertar/Melhorar Samuel Santos !!!!!): data
            apresenta a coluna da base de dados
341         ela retornada (como um vetor de vetores????) e medida para
            que n o d out of range
            e ent o apresentada passo por passo

343         o row n o varia de acordo com um vetor

345         Procurar descobrir ou como usar o row[0] ou tirar a descri o
            sql de data

347         '''

349         for i in range(len_title):
351             btn = Button(text="Button " + str(title[i]),

```

```

353         size=(810, 50),
        size_hint=(None, None),
355         background_color = (0,0,0,1),
        font_size =(36),
357         on_press=self.Press_auth) # <<<<<<<<<<<<<<<<<<

359     layout.add_widget(btn)

361     btn_volta = Button(text="Volta",
        pos_hint = ({ "left": 1, "bottom": 1}),
363         size=(810, 700),
        size_hint=(None, None),
365         background_color=(0, 0, 0, 1),
        font_size=(36),
367         on_press=self.Press_auth) # <<<<<<<<<<<<<<<<<<

369     layout.add_widget(btn_volta)

371     root = ScrollView()
    root.add_widget(layout)
373     self.add_widget(root)

375     def Press_auth(self, instance):
        screen_manager.transition = NoTransition()
377         screen_manager.current = 'screen_one'
        print(str(instance))

379

381 # The ScreenManager controls moving between screens
    screen_manager = ScreenManager()

383

385 # Add the screens to the manager and then supply a name
# that is used to switch screens
    screen_manager.add_widget(ScreenOne(name="screen_one"))
387     screen_manager.add_widget(ScreenTwo(name="screen_two"))
    screen_manager.add_widget(ScreenThree(name="screen_three"))

389

391 class KivyTut2App(App):

393     def build(self):
395         return screen_manager

397

sample_app = KivyTut2App()

399
```

```
401 # ----- SENSOR PIR -----
403 def MOTION(PIR_PIN):
405     print ("Motion Detected");
407
408     # Enviando o email
409     server = smtplib.SMTP('smtp.gmail.com:587')
410     server.starttls()
411     server.login(remetente, senha)
412     server.sendmail(remetente, destinatario, msg)
413     server.quit()
414     time.sleep(7)
415
416 print ("PIR work");
417 time.sleep(2)
418 print ("Ready");
419
420 try:
421     GPIO.add_event_detect(PIR_PIN, GPIO.RISING, callback=MOTION)
422
423 except KeyboardInterrupt:
424     print ("Quit");
425     GPIO.cleanup()
426
427 if __name__ == '__main__':
428     if os.environ.get("WERKZEUG_RUN_MAIN") != 'true':
429         threading.Thread(target=start_app).start()
430         sample_app.run()
```


APÊNDICE F – Todo HTML

```
<!DOCTYPE html>
2 <html lang="en">

4 <head>
    <meta charset="utf-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
8    <meta name="author" content="">

10    <title>MUSK</title>

12    <!-- css -->
    <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" />
    <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" />
    <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" />
    <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" />
    <link href="{{ url_for('static', filename='css/todo_style.css') }}" rel
        ="stylesheet">

22    <!-- boxed bg -->
24    <link id="bodybg" href="bodybg/bg1.css" rel="stylesheet" type="text/css"
        />
    <!-- template skin -->
26    <link id="t-colors" href="{{ url_for('static', filename='color/default.
        css') }}" rel="stylesheet">

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

```

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

```

```

68         {% for todo in incomplete %}
        <li><span><a href="{{ url_for('complete', id=todo.id) }}">
          </a> </span>
          <span class="list"><strong>{{ todo.title }}, {{ todo.start
            }} </strong></span>
        </li>
72     {% endfor %}
    </ul>

74
    </div>
76 </div>

78
80 </div>

82 <div class="col-lg-6">
    <div class="form-wrapper">
    <div class="wow fadeInRight" data-wow-duration="2s" data-wow-
84         delay="0.2s">

        <div class="panel panel-skin">
86         <div class="panel-heading">
            <h3 class="panel-title"><span class="fa fa-pencil-square-
              o"></span> Adicione um Compromisso </h3>
88         </div>
            <div class="panel-body">
            <form role="form" class="lead" action="{{ url_for('add')
              }}" method="POST">
            <div class="row">
92                <div class="col-xs-6 col-sm-6 col-md-6">
                    <div class="form-group">
94                        <label>Título</label>
                        <input type="text" name="todoitem" class="form-
                          control input-md">
96                    </div>
                </div>
                <div class="col-xs-6 col-sm-6 col-md-6">
98                    <div class="form-group">
                        <label>Data</label>
100                        <input type="date" name=
                          ="date" id="phone"
                          class="form-con$">
102                    </div>
                </div>
            </div>
104        </div>
    </div>

```

```

106         <div class="row">
107             <div class="col-xs-6 col-sm-6 col-md-6">
108                 <div class="form-group">
109                     <label style="color:
110                         white;" >.</label>
111                 </div>
112             </div>
113             <div class="col-xs-6 col-sm-6 col-md-6">
114                 <div class="form-group">
115                     <label style="color:
116                         white;" >.</label>
117                 </div>
118             </div>
119         </div>
120         <input type="submit" value="Submit" class="btn btn-skin
121             btn-block btn-lg">
122     </form>
123 </div>
124 </div>
125
126 </div>
127 </div>
128
129
130 <div class="row">
131     <div class="col-lg-6" style="margin-right: 300px;">
132     <div class="wow fadeInDown" data-wow-offset="0" data-wow-delay="
133         0.1s">
134     <h2 class="h-ultra"></h2>
135     </div>
136     <div class="wow fadeInUp" data-wow-offset="0" data-wow-delay="0.1
137         s">
138     <h4 class="h-light">Tarefas <span class="color">Finalizadas:</
139         span></h4>
140     </div>
141     <div class="well well-trans">
142     <div class="wow fadeInRight" data-wow-delay="0.1s">
143
144         <ul class="lead-list">
145             {% for todo in complete %}
146             <li><span class="list"><
148                 strong>{{ todo.title }}</strong></span></li>
149             {% endfor %}
150         </ul>

```

```

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

```

APÊNDICE G – Todo Python

```
1 from flask import Flask, render_template, request, redirect, url_for
2 from flask_sqlalchemy import SQLAlchemy
3 from datetime import datetime
4
5 import schedule
6 import time
7
8
9 app = Flask(__name__)
10
11 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///home/pi/web-server/todo
    .db'
12
13 db = SQLAlchemy(app)
14
15
16 class Todo(db.Model):
17     id = db.Column(db.Integer, primary_key=True)
18     title = db.Column(db.String(200))
19     start = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
20     complete = db.Column(db.Boolean)
21
22
23 @app.route('/')
24 def index():
25     incomplete = Todo.query.filter_by(complete=False).all()
26     complete = Todo.query.filter_by(complete=True).all()
27
28     return render_template('todotestsite.html', incomplete=incomplete,
29                             complete=complete)
30
31 @app.route('/add', methods=['POST'])
32 def add():
33     todo = Todo(title=request.form['todoitem'], start=datetime.strptime(
34         request.form['date'], '%Y-%m-%d'), complete=False)
35     db.session.add(todo)
36     db.session.commit()
37
38     return redirect(url_for('index'))
39
40 @app.route('/complete/<id>')
```

```
39 def complete(id):  
  
41     todo = Todo.query.filter_by(id=int(id)).first()  
    todo.complete = True  
43     db.session.commit()  
  
45     return redirect(url_for('index'))  
  
47 if __name__ == '__main__':  
    app.run(host='0.0.0.0', threaded=True, debug=True)
```