FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Eduardo Moriggi Filho

Controle de acesso por reconhecimento facial e controle de ambientes

Americana, SP
2018

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Eduardo Moriggi Filho

Controle de acesso por reconhecimento facial e controle de ambientes

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Kleber de Oliveira Andrade.

Área de concentração: Inteligência artificial.

Americana, SP

2018

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

M853c MORIGGI FILHO, Eduardo

Controle de acesso por reconhecimento facial e controle de ambientes. / Eduardo Moriggi Filho. – Americana, 2018.

80f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kleber de Oliveira Andrade

1 Inteligência artificial 2. Biometria 3. Internet das coisas I. ANDRADE, Kleber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 007.52
681.6
681.518

Eduardo Moriggi Filho

Controle de acesso por reconhecimento facial e controle de ambientes

Trabalho de Graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/AMERICANA.

Área de concentração: Inteligência artificial.

Americana, 06 de dezembro de 2018.

Banca **nadora:**

Kleber de Oliveira Andrade (Presidente)
Doutor
Faculdade de Tecnologia de Americana

Murilo Fujita (Membro)
Mestre
Faculdade de Tecnologia de Americana

Clerivaldo José Roccia (Membro)
Mestre
Faculdade de Tecnologia de Americana

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, pois ele me deu forças para a realização deste trabalho.

Agradeço a minha família, pelo apoio e motivação que me fez persistir na elaboração deste trabalho.

À minha namorada, Pamela Klava Senna Patrício pelo apoio durante todo o desenvolvimento deste trabalho.

Ao meu orientador, Prof. Dr. Kleber de Oliveira Andrade pela ajuda na estruturação deste trabalho.

Agradeço também a todos que me ajudaram direta ou indiretamente com o trabalho.

RESUMO

O reconhecimento facial está presente no dia a dia da sociedade, seja através de biometria facial, aplicativos de entretenimento, análise forense, entre outros. A segurança é um dos assuntos mais discutidos na sociedade, por este motivo, desenvolver tecnologia para garantir a segurança é essencial. Este projeto vem para solucionar um problema de segurança no que diz respeito ao controle de acesso, que através do reconhecimento facial controlará o acesso a locais restritos, permitindo a entrada somente de pessoas previamente autorizadas com sua face cadastrada no sistema. É fascinante a aplicação do reconhecimento facial, algo que os seres humanos fazem de forma relativamente fácil, a uma máquina. Porém, apesar de o reconhecimento facial ser uma tarefa relativamente fácil para os seres humanos, para as máquinas o desafio parece ser muito maior, pois, apesar de já haver algoritmos que trabalhem com alta precisão, o rosto humano tem uma grande diversidade de caraterísticas e quando aplicado a algoritmos de reconhecimento facial em tempo real, está submetido a variações de luz, posição do rosto etc. Por esse motivo os algoritmos trabalham melhor em ambientes controlados. Os procedimentos para o desenvolvimento da aplicação para reconhecimento facial são basicamente, detectar faces através de um modelo em cascata conhecido como técnica de Viola Jones, a seguir, deve-se extrair um conjunto de dados contendo faces através da câmera RGB do Kinect, treiná-los e classificá-los de forma que seja gerada uma saída de um vetor de características, que quando comparado à outras faces é obtido um valor de confiabilidade, podendo definir se a face pertence ou não à pessoa de interesse. A parte física deste projeto foi elaborada com base no Rasbperry Pi versão 3B Plus, em conjunto com dispositivos e sensores, sendo todos alojados dentro de uma caixa metálica de forma compacta e segura. O módulo *web* deste projeto aborda o controle de ambientes, que torna possível a visualização e controle de dados de temperatura, umidade e acesso. Além de tornar possível o controle de dispositivos através do computador, *tablet*, celular ou qualquer outro dispositivo que possua um navegador de internet.

Palavras Chave: Inteligência artificial. Biometria. Internet das coisas.

ABSTRACT

Facial recognition is present in everyday society, whether through facial biometrics, entertainment applications, forensic analysis, among others. Security is one of the most discussed issues in society, so developing technology to ensure security is essential. This project comes to solve a security problem with regard to access control, which through facial recognition will control access to restricted locations, allowing the entry of only previously authorized persons with their face registered in the system. It's fascinating to apply facial recognition, something humans do relatively easily to a machine. However, although facial recognition is a relatively easy task for humans, for machines the challenge seems to be much greater, since, although there are already algorithms that work with high precision, the human face has a great diversity of characteristics and when applied to facial recognition algorithms in real time, is subject to variations of light, face position etc. For this reason algorithms work best in controlled environments. The procedures for developing the application for facial recognition are basically to detect faces through a cascade model known as the Viola-Jones technique, then a data set containing faces must be extracted through the Kinect RGB camera, training and classify them in such a way that an output of a feature vector is generated, which when compared to other faces is obtained a reliability value, being able to define whether the face belongs or not to the person of interest. The physical part of this project was developed based on the Rasbperry Pi version 3B Plus, together with devices and sensors, all housed inside a metal box in a compact and safe way. The web module of this project addresses the control of environments, which makes possible the visualization and control of temperature, humidity and access data. In addition to making it possible to control devices through a computer, tablet, mobile phone or any other device that has a web browser.

Keywords: Artificial intelligence. Biometry. Internet of things.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application programming interface
BSD	Berkeley Software Distribution
CMOS	Complementary Metal Oxide Semiconductor
GUI	Graphical user interface
GPIO	General-purpose Input/Output
HDMI	High-Definition Multimedia Interface
ID	Identifier
IR	Infrared
LCD	Liquid Crystal Display
LPDDR2	Low Power Double Data Rate
PIR	Passive infrared sensor
RGB	Red, green, blue
SD	Secure Digital
SDRAM	Synchronous Dynamic Random Access Memory
SIFT	Scale-invariant feature transform
SLC	Serial Clock
SURF	Speeded up robust features
SDA	Serial Data
USB	Universal Serial Bus
UX	User Interface
WSGI	Web Server Gateway Interface
3D	Three-dimensional

1 INTRODUÇÃO

O reconhecimento facial e a automação utilizando microcontroladores estão em alta no mercado de tecnologia, isso foi levado em consideração na hora de pensar no desenvolvimento da aplicação, onde foram desenvolvidos dois módulos. O modulo de controle de acesso por reconhecimento facial foi desenvolvido para funcionar com uma *interface desktop*, já o módulo de controle de ambientes, possui uma *interface web*, onde é possível controlá-los através de qualquer dispositivo que possua um navegador de internet. O módulo de controle de acesso por reconhecimento facial, proporciona um ambiente controlado, permitindo somente a entrada de pessoas autorizadas, ou seja, que tenham seus rostos previamente fotografados e treinados pelo algoritmo de reconhecimento facial, onde o nome é associado a imagem possuindo uma identidade única, isso torna o ambiente inteligente e seguro. O projeto foi pensado para que sua implementação seja feita em locais onde se deseja um acesso restrito, por exemplo, em sala de servidores, onde além do controle de acesso ser importante, as condições de clima como umidade e temperatura são fundamentais para garantir o correto funcionamento das máquinas, prevenindo uma possível condensação causada pela alta umidade do ar, ou às altas temperaturas que podem prejudicar a vida útil dos equipamentos.

A fim de comparação das funcionalidades dos *softwares* de reconhecimento facial e controle de ambientes, serão comparadas as funcionalidades de sistemas populares e pertencentes a grandes empresas de tecnologia como Google, Amazon, Microsoft etc.

- O **Google Fotos** é um aplicativo desenvolvido pela Google (2018) que possui algoritmos para detecção e reconhecimento de imagens, o serviço reúne imagens que contenham a mesma pessoa em uma lista automaticamente. Porém, não é capaz de fazer a distinção de idade e gênero como outros aplicativos.
- O **Amazon Rekognition** é uma API desenvolvida por cientistas de visão computadorizada da Amazon (2018), e segundo a própria desenvolvedora ele oferece reconhecimento facial preciso, sendo possível ser utilizado através de imagens vídeos e até mesmo em tempo real. Além do reconhecimento facial o serviço é capaz de identificar pessoas, objetos, conteúdos inadequados etc.

- O **Microsoft Face** é uma API desenvolvida pela Microsoft (2018) que dispõe de uma variedade de recursos específicos para a análise facial. Ela é capaz de detectar e comparar faces humanas, organizar imagens em grupos se baseado na similaridade das imagens faciais. Fornece também uma verificação facial onde através dela é obtida uma pontuação de confiança sobre a probabilidade de que uma face pertença a uma pessoa, entre outros recursos.

Os aspectos que serão comparados em relação ao *software* desenvolvido neste projeto serão a detecção facial, reconhecimento de face, reconhecimento de idade e gênero, rastreamento de várias faces, tempo real, controle de dispositivos, verificação de face. A Tabela 1 foi elaborada para demonstrar as principais diferenças entre o sistema desenvolvido neste trabalho. Note que, a sigla S1 destacada representa o sistema desenvolvido neste trabalho, enquanto S2, S3, S4, representam, respectivamente, Google Fotos, Amazon Rekognition e Microsoft Face:

Tabela 1 – Comparativo de funcionalidades do sistema

Funcionalidade	**S1**	**S2**	**S3**	**S4**
Detecção de Face	**X**	X	X	X
Reconhecimento de Face	**X**	X	X	X
Idade e Gênero			X	X
Rastreamento de Várias Faces	**X**	X	X	X
Tempo Real	**X**		X	X
Controle de Dispositivos	**X**			
Verificação de Face	**X**		X	X

Fonte: Elaborado pelo autor

Este trabalho tem como objetivo, obter um sistema funcional que seja capaz de coletar dados de sensores e colocá-los em gráficos e tabelas, possibilitando um controle simples e intuitivo das condições de temperatura, umidade e de acesso. Que também que possua controle de acesso por reconhecimento facial em tempo real sem travamentos, com assertividade e que seja fácil de utilizar.

2 PROJETO DO SISTEMA

O *software* conta com dois módulos, um sistema *desktop* para o acesso por reconhecimento facial e o módulo *web* para controle de ambientes. A versão *desktop* é gerenciada pelo administrador e o usuário tem permissão apenas para ter seu rosto cadastrado e posteriormente reconhecido pelo sistema. O módulo *web* serve para que o administrador tenha acesso as condições do ambiente, relatório de acesso e controle de dispositivos de forma remota pelo navegador de internet. Este capítulo detalha o processo do desenvolvimento com a elaboração dos requisitos funcionais e não funcionais do sistema, ferramentas utilizadas, diagrama de caso de uso e fluxograma.

2.1 Levantamento de Requisitos

A engenharia de requisitos (RE – *Requirements Engineering*) é o processo de descobrir, analisar, documentar e verificar requisitos de um sistema. Um requisito pode ser definido como uma descrição dos serviços fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007). Tradicionalmente, os requisitos são divididos em dois tipos: requisitos funcionais e requisitos não funcionais.

2.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, definem a funcionalidade desejada do *software* (SOMMERVILLE, 2007). A Tabela 2 apresenta os requisitos funcionais deste projeto.

Tabela 2 – Requisitos funcionais do projeto

Identificação	Requisito Funcional	Prioridade
RF001	Reconhecimento de face	Alta
RF002	Detecção de face	Alta
RF003	Verificação de face	Média
RF004	Tempo real	Alta
RF005	*Stream* de vídeo da *webcam*	Média
RF006	Controle de dispositivos	Alta
RF007	Controle de temperatura e umidade	Alta
RF008	Armazenamento de dados de temperatura e umidade	Alta
RF009	Armazenamento de dados de acesso	Alta
RF010	Assistência por voz no controle de acesso	Média
RF011	*Website* deve ser responsivo	Alta
RF012	Dados de temperatura e umidade devem ser exportados via API do Plotly	Média

Fonte: Elaborado pelo autor

2.1.2 Requisitos Não Funcionais

"Os requisitos não funcionais são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema" (SOMMERVILLE, 2007). A Tabela 3 apresenta os requisitos não funcionais deste projeto.

Tabela 3 – Requisitos não funcionais do projeto

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	*Website* se adapta a largura de tela de qualquer dispositivo	Usabilidade	Essencial
RNF002	Reconhecimento facial assertivo	Confiabilidade	Essencial
RNF003	Reconhecimento facial em tempo real sem travamentos	Desempenho	Essencial
RNF004	Interface do *website* é limpa e intuitiva	Usabilidade	Essencial
RNF005	Assistente de voz que facilita uso do acesso por reconhecimento facial	Usabilidade	Essencial
RNF006	Módulo *web* e *desktop* desenvolvidos em sua maior parte em Python	Padrões	Essencial
RNF007	Utiliza a última versão do Raspberry Pi	*Hardware* e *Software*	Essencial

Fonte: Elaborado pelo autor

2.2 Recursos e Ferramentas

Nesta seção serão abordados os modelos, ferramentas de programação, bibliotecas e *frameworks* que foram utilizados para o desenvolvimento do sistema web e *desktop* deste projeto. Para este projeto foram escolhidas ferramentas modernas e otimizadas para a sua utilização com o Raspberry Pi ofereça um bom desempenho e confiabilidade.

2.2.1 SQlite 3

O SQLite é um sistema de gestão de base de dados relacional e uma opção popular para criar banco de dados de sites de pequeno a médio porte. De acordo com o próprio site do SQlite (2018), ele é um banco excelente para “Internet das Coisas”, pois o possui código de tamanho pequeno e é eficiente quando se trata de gerenciamento de memória, espaço em disco e largura de banda de disco, além de ser altamente confiável. O sistema gerenciador de banco de dados SQlite 3 não precisa de um administrador de banco de dados, pois não possui um processo de servidor separado e pode armazenar informações em arquivos de disco comuns. Devido a todas essas características, tornou-se o modelo ideal para aplicação neste projeto que não requer grande quantidade de armazenamento e precisa de um banco de dados leve, que não sobrecarregue o Raspberry Pi. Outro ponto

importante é que o código do SQlite 3 é de domínio público, portanto, é livre e gratuito para qualquer finalidade.

2.2.2 Raspberry Pi

O Raspberry Pi é um pequeno computador, desenvolvido pela Raspberry Pi Foundation (2018), que apesar do tamanho, oferece um poder de processamento excelente, com um processador quad-core de 64 bits de 1.4 GHz, rede sem fio de banda dupla com padrão IEEE 802.11.b/g/n/ac, Bluetooth 4.2 e 1GB de memória LPDDR2 SDRAM, sendo um dos melhores da categoria disponíveis no mercado. O Raspberry Pi possui uma ótima eficiência energética, pois de acordo com o fabricante Raspberry Pi Foundation (2018), é necessário apenas um carregador de 5V a 2.5A, para que seja possível utilizá-lo. O Raspberry Pi possui diversas conexões, entre elas estão, uma entrada micro USB utilizada como fonte de alimentação para o Raspberry Pi, *slot* para ligação de uma câmera, pinos para utilização de energia pelo cabo ethernet (POE - *power over ethernet*), conexão Fast-Ethernet, entrada para cartão de memória micro SD, HDMI, conector de áudio e vídeo, pinos GPIO. É necessário a utilização de um cartão SD para a instalação do sistema operacional, pois é através do cartão micro-SD que é realizado o armazenamento de todos os dados no Raspberry Pi, incluindo o sistema operacional e arquivos.

2.2.3 OpenCV 3

OpenCV é uma biblioteca de processamento de imagens originalmente desenvolvida pela Intel (2000), e desenvolvido em C/C++. Foi projetado para ser eficiente computacionalmente e com foco em aplicativos em tempo real. O OpenCV possui interfaces C++, é uma biblioteca multiplataforma e oferece suporte a diversos sistemas operacionais como, por exemplo, Windows, Linux, Mac OS, iOS e Android. É uma biblioteca *open source* e não requer que a aplicação desenvolvida seja de código aberto, pode ser usada para fins como arte interativa, inspeção de minas ou através de robótica avançada.

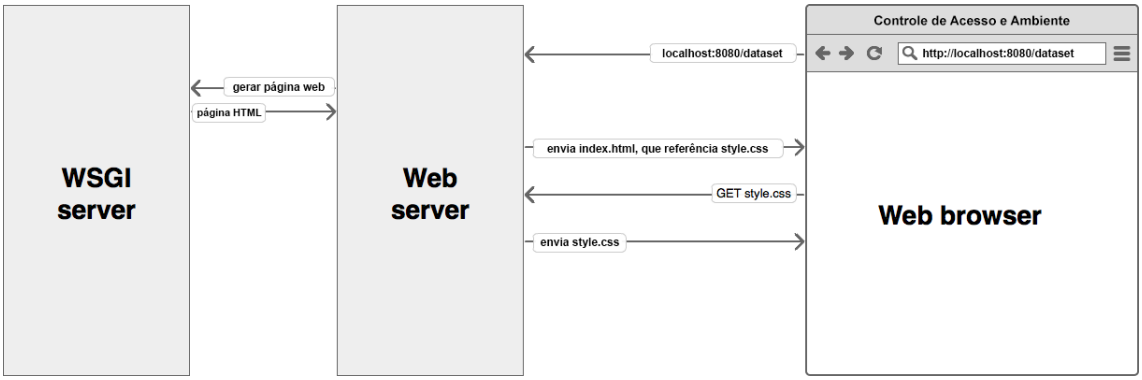
2.2.4 Kinect

O Kinect (2018) é um *hardware* que contém uma câmera RGB, um projetor laser IR, um sensor CMOS IR, um servo para ajustar a inclinação do dispositivo e uma matriz de microfone. O Kinect foi lançado em 4 de novembro de 2010 e logo após o lançamento desenvolvedores já começaram a explorar o *hardware* e criar aplicações personalizadas utilizando seus componentes, como a câmera de profundidade 3D, o que criou muitas oportunidades para a computação, multimídia e outras áreas.

2.2.5 Flask

Flask é um micro *framework web* escrito em Python e baseado na biblioteca WSGI Werkzeug (WSGI - *Web Server Gateway Interface*) e na biblioteca de Jinja2. Ele é chamado de micro *framework* pois possuí um núcleo simples, porém que pode ser estendido. Por exemplo, ele não possui muitos componentes próprios para validação de formulários etc., porém, é possível a integração de muitas bibliotecas de terceiros para esses e outros fins. O uso do Flask neste projeto foi motivado por ser um modelo simples para desenvolvimento web, onde pode-se utilizar a linguagem Python. O Flask está disponível sob os termos da Licença BSD. O WSGI é uma especificação para uma interface simples e universal entre servidores *web* e aplicações *web* ou *frameworks* para a linguagem de programação Python. Através do dimensionamento o WSGI pode atender milhares de solicitações de conteúdo dinâmico de uma só vez. Servidores WSGI manipulam solicitações de processamento do servidor da *web* e decidem como comunicar essas solicitações ao processo de uma estrutura de aplicativo. Assim, se obtém uma segregação de responsabilidades tornando o dimensionamento eficiente do tráfego da web. O padrão WSGI Foi originalmente especificada na PEP 333, por Philip J. Eby, (2003). A partir de setembro de 2010, o WSGI v1.0 foi substituído pelo PEP 3333, que define o padrão WSGI v1.0.1.

Figura 1 – Funcionamento do micro framework Flask



Fonte: Elaborado pelo autor.

A instalação do Flask exige algumas dependências que são instaladas automaticamente, como:

- O Werkzeug implementa o WSGI, a interface padrão do Python entre aplicativos e servidores.
- Jinja é uma linguagem de modelo que processa as páginas que seu aplicativo serve.
- O MarkupSafe vem com o Jinja. Ele age como um filtro evitando ataques de injeção de código.
- ItsDangerous é implementado para atuar na segurança dos dados, garantindo sua integridade, sendo usados para proteger o *cookie* de sessão do Flask.

2.3 Detecção facial

Os algoritmos de detecção facial funcionam a partir características (*features*) faciais. Assim, o algoritmo procura por essas características nas fotos, extraindo os dados necessários para poder determinar se objeto de interesse é uma face ou não. Porém, é possível se obter uma alta porcentagem de exposição em uma face na foto e ainda assim a detecção falhar se as características de interesse, estiverem escondidos ou forem manipulados, como a posição, expressão facial, iluminação etc. Portanto, pensar no sucesso da detecção facial como se ela dependesse de uma porcentagem mínima de exposição do rosto na foto não é uma abordagem

realística. No artigo Eigenfaces vs Fishfaces, de Peter N. Belhumeur et al. (1997) do Departamento de Engenharia Elétrica da Universidade Yale, é discutido uma técnica mais robusta para solucionar esses problemas, onde faz uma comparação interessante com Eigenfaces e Fishfaces.

2.3.1 Técnica de Viola-Jones

Na técnica de Viola e Michel Jones (2001) é utilizada a ideia de Janelas de Processamento que extrai características da imagem e seleciona o mínimo de características a serem treinadas com uma implementação de Aprimoramento Adaptativo (AdaBoost - *Adaptative Boosting*) e depois combina os classificadores simples com uma estrutura em cascata. O classificador em cascata é caracterizado por características Haar-like (*Haar-like features*). O classificador em cascata é chamado assim porque combina vários classificadores fracos. Os classificadores fracos levam esse nome pois sozinho não pode classificar a imagem, porem uma soma ponderada desses classificadores é capaz de formar um classificador mais forte, que é chamado de método de reforço. Segundo o artigo de Paul Viola e Michael Jones (2001), até 200 recursos fornecem detecção com 95% de precisão. A principal vantagem deste método é sua baixa complexidade computacional e sua paralelização, sendo uma das técnicas mais utilizadas na indústria ou no meio científico para detecção de faces ou objetos.

2.3.2 AdaBoost

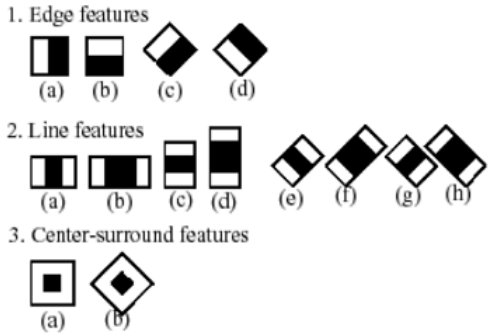
O Aprimoramento Adaptativo (AdaBoost - abreviação de *Adaptive Boosting*), é um meta-algoritmo de aprendizado de máquina desenvolvido por Yoav Freund e Robert Schapire (1997). A saída dos outros algoritmos de aprendizado que são chamados de “fracos”, pois sozinhos não podem classificar uma imagem, são combinados em uma soma ponderada dos classificadores “fracos”, onde passa a ser um classificador forte por unir e ponderar diversas características, assim, gerando no final a saída de um classificador impulsionado e eficiente. O Aprimoramento Adaptativo pode ser utilizado em conjunto com outros tipos de algoritmos de

aprendizado para melhorar o desempenho. Neste projeto foi utilizado para se tirar proveito da característica do algoritmo de selecionar apenas os recursos conhecidos para melhorar o poder preditivo do modelo, de acordo com o Efeito Hughes. Assim, reduzindo a dimensionalidade e surtindo uma melhora significável no tempo de execução, já que recursos irrelevantes não são computados. No modelo de detecção de objetos de Viola-Jones (2001), que foi utilizado nesse projeto, segundo o documento pode haver 162.336 recursos do Haar em uma janela de 24x24 pixels, já com a utilização do Aprimoramento Adaptativo o documento diz que até 200 recursos fornecem detecção com 95% de precisão. Sua configuração final tinha cerca de 6000 recursos, contra os 162.336 recursos necessários sem a utilização do Aprimoramento Adaptativo.

2.3.3 Haar Cascade

O algoritmo Haar Cascade usa as respostas a uma série de filtros simples para classificar as regiões de uma imagem como uma face ou não como uma face. Os filtros são chamados de filtros Haar e são calculados tomando a soma dos *pixels* em um número de retângulos, multiplicando cada soma por um peso e adicionando os resultados.

Figura 2 – Filtros utilizados para a extração de características faciais.

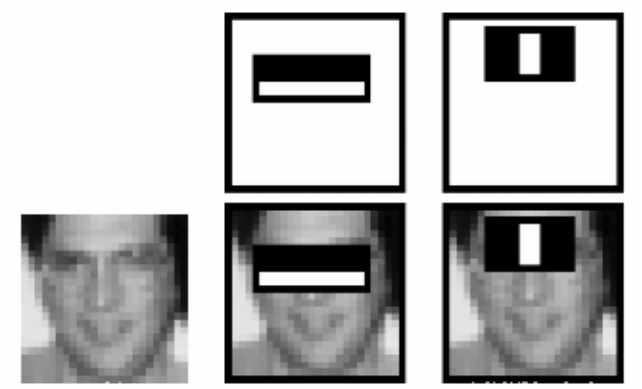


Fonte: OpenCV (2018).

De acordo com o Site do OpenCV temos que o Método Haar Cascade é um método eficaz de detecção de objetos proposto por Paul Viola e Michael Jones (2001) em seu artigo, "Rapid Object Detection using a Boosted Cascade of Simple

Features". Ele se trata de uma abordagem baseada em *machine learning* (aprendizado de máquina), onde através de característica é possível definir padrões, tornando-se possível diferenciar texturas e classificar os objetos que se deseja reconhecer.

Figura 3 – Conjunto de exemplos de características faciais



Fonte: OpenCV (2018).

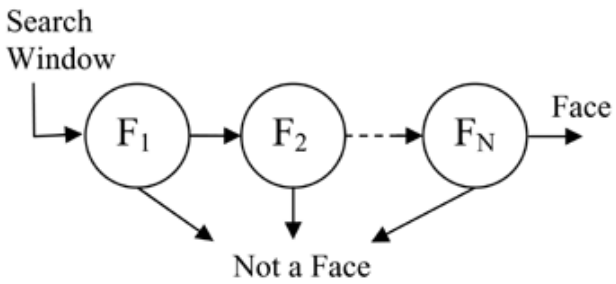
Neste projeto está sendo utilizado um modelo em cascata previamente treinado e específico para detecção frontal de faces criado por Lienhart et. Al. (2002). Segundo o artigo de Lienhart et. Al. (2002), chamado "*Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection*", o treinamento da cascata utilizou 5000 imagens positivas e 3000 imagens negativas que segundo o autor, foi o suficiente para chegar perto de seu poder de representação.

A função de cascata é treinada a partir de amostras contendo imagens positivas e muitas imagens negativas:

- Amostra positiva: São imagens do objeto de interesse, contém diferentes representações do objeto a ser detectado sob diferentes perspectivas, condições de iluminação, tamanhos etc.
- Amostra negativa: São imagens que não contém o objeto de interesse.
- O número de imagens necessárias depende de uma variedade de fatores, incluindo a qualidade das imagens, o objeto de interesse e a potência de processamento.

No modelo cascata, ao invés de aplicar todos os recursos em uma janela, são agrupados os recursos em diferentes estágios de classificadores e aplicados um por um, onde normalmente, nos primeiros estágios conterão um número muito menor de recursos, aumentando conforme a detecção avança de estágio. Assim, no caso de uma janela falhar logo primeiro estágio, ela é descartada imediatamente e todos os recursos restantes são desconsiderados. Caso complete o primeiro estágio é aplicado o segundo estágio dos recursos e assim por diante, até o final do processo e as janelas se esgotarem. A janela que passa por todos os estágios é uma região de face.

Figura 4 – Exemplo de janelas de recursos



Fonte: OpenCV (2018).

2.4 Reconhecimento facial

Os cérebros humanos estão muito acostumados a reconhecer faces, essa habilidade é tamanha, que são capazes de identificar rostos em diversos objetos, onde pela sua simetria são expressadas características de um rosto humano, assim, o cérebro humano é capaz de aplicar o reconhecimento facial com certa facilidade, porém, o reconhecimento facial desenvolvido através de um computador pode ser muito mais trabalhoso, considerando-se uma grande quantidade de fatores. O ciclo que deve ser completo para que seja possível o reconhecimento facial é parecido entre os humanos e computadores, possuindo quatro etapas básicas. Primeiramente deve ser realizada a obtenção das imagens de faces, de onde serão extraídas as características faciais, como os olhos, nariz, boca e tom de pele. Quanto mais dados de características forem coletados da face, mais preciso será o reconhecimento facial. As etapas basicamente são:

- Detecção de rosto: Analisando a imagem e encontrando um rosto nela.
- Coleta de dados: Extrair características únicas da face de interesse para diferenciá-la.
- Comparação de dados: Comparação de recursos exclusivos a todos os recursos de todas as faces já conhecidas.
- Reconhecimento facial: Determinar a identidade da face conforme suas características únicas.

Figura 5 – Imagens que lembram rostos humanos encontradas em objetos

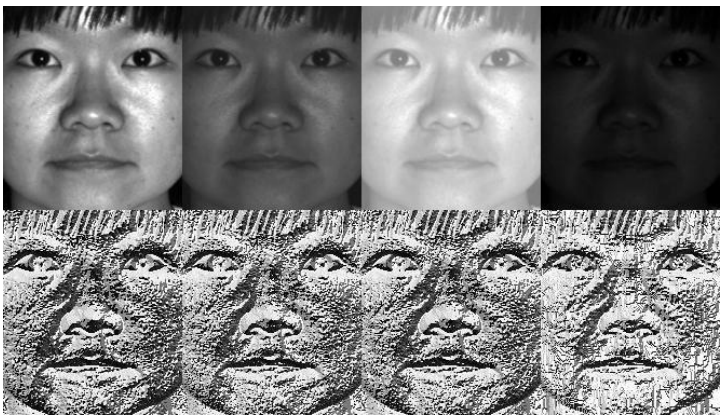
Fonte: Google Imagens (2018).

Para que o computador seja capaz de reconhecer rostos, neste projeto foi desenvolvido um sistema com a finalidade de ser um controlador de acesso a ambientes restritos, utilizando o reconhecimento facial, onde é utilizada uma implementação de *machine learning*, utilizando o Python 2.7, a biblioteca OpenCV 3 e outras bibliotecas. A biblioteca Numpy é necessárias para a execução do OpenCV, a biblioteca Pickle é utilizada para gerar etiquetas para a identificação dos rostos contendo um ID único e nome correspondente para cada usuário, a biblioteca OpenKinect é utilizada para a utilização do *hardware* Kinect como entrada de dados de imagem, a biblioteca SQlite 3 é utilizada como base de dados e foi escolhida por ser leve e compacta, oferecendo um bom desempenho e capacidade suficiente para o processamento dos dados de log de acesso, temperatura e umidade no Raspberry Pi. A biblioteca Text to Speech, utiliza uma API do Google para sintetização de voz, onde é possível fazer com que a assistente de voz leia uma variável de texto e isso foi usado para orientar o usuário no momento do reconhecimento facial.

2.4.1 Local Binary Patterns

O descritor visual *Local Binary Patterns* ou em português “Padrões Binários Locais”, referidos como LBP, são utilizados para classificação em visão computacional, foram originalmente propostos por Ojal et al. (1996). Porém, o LBP se popularizou após Ojala et al. (2002) publicarem o artigo “*Multiresolution Grayscale e Rotation Invariant Texture Classification with Local Binary Patterns*”. O algoritmo principal LBP foi usado para determinar as texturas de imagem, onde é calculada uma representação local da textura constituída pela comparação de cada *pixel* com sua vizinhança. Ele possuiu velocidade de cálculo muito rápida e tem uma invariância de rotação e gradação. Como o LBP é um descritor visual, ele também pode ser usado para tarefas de reconhecimento de rosto. Usando o LBP combinado com histogramas, os chamados Histogramas de Padrões Binários Locais (LBPH – *Local Binary Patterns Histogram*), podemos representar as imagens faciais com um simples vetor de dados. São usados também para uma ampla gama de aplicações que vão desde detecção de face como abordado no artigo de B. Froba and A. Ernst (2004), “*Face detection with the modified census transform*”, reconhecimento de expressão facial como no artigo de Caifeng Shan, Shaogang Gong, and Peter W. McOwan (2009), “*Facial expression recognition based on Local Binary Patterns: A comprehensive study*”, entre outros, a fim de construir poderosos sistemas de detecção de objetos visuais. Ao comparar um grupo de imagens contendo uma face com suas respectivas imagens de padrões binários locais, pode-se ver que as faces LBP são pouco afetadas por mudanças nas condições de luz:

Figura 6 – Demonstração do efeito da luz na extração de LBP em imagens



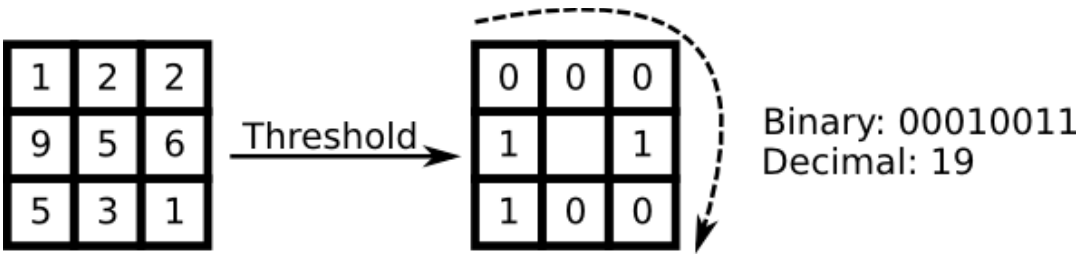
Fonte: OpenCV (2018).

A extração dos vetores de histograma LBP são capazes de reconhecer padrões em texturas de imagens, para que seja possível classificá-las, basicamente o algoritmo deve seguir os seguintes passos:

Deve-se pegar na foto a área de interesse, ou seja, onde há uma face e transformá-la em escala de cinza.

Para cada *pixel* na imagem em escala de cinza deve-se selecionar a vizinhança ao redor do *pixel* central para que possa ser calculado o valor de LBP para este mesmo *pixel* central. Para a realização do cálculo, deve-se pegar o pixel central e o limitar contra sua vizinhança de 8 pixels, no sentido horário ou anti-horário, desde que isso siga um padrão para todos os pixels da imagem e conjunto de dados.

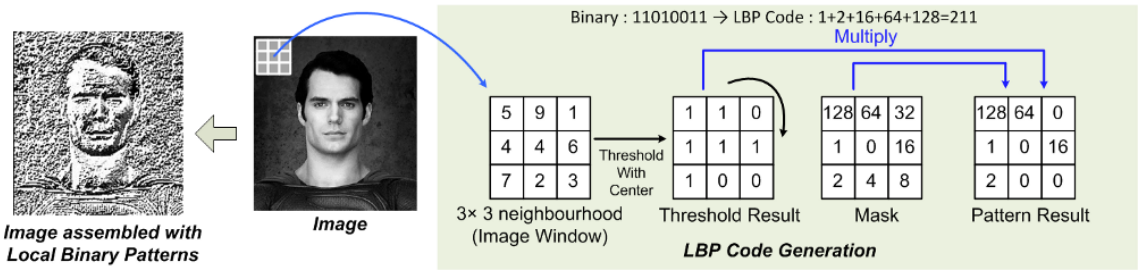
Figura 7 – Cálculo do pixel central do bloco



Fonte: OpenCV (2018).

A seguir verifica-se a intensidade do *pixel* central e caso seja maior ou igual a seu vizinho, então o valor é definido como 1, caso contrário, é definido como 0. Caso o pixel esteja nas bordas os vizinhos não contidos no bloco são definidos como 0. Assim, com 8 pixels ao redor, é construído um conjunto de 8 dígitos binários. São possíveis ser feitas 2^8 combinações, ou seja, 256 combinações possíveis de blocos de códigos LBP, tendo um valor mínimo de 0 e um valor máximo de 255.

Figura 8 – Demonstração de como é feito o cálculo do LBP

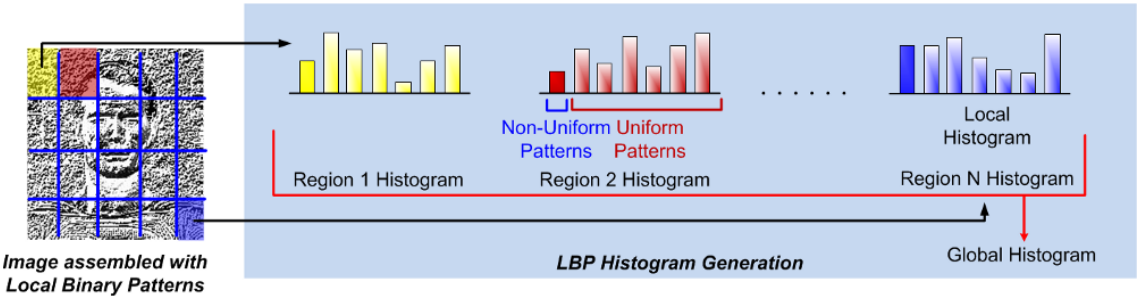


Fonte: Medium (2017).

O bloco com seus 8 dígitos binários é convertido em valor decimal e inserido em uma matriz LBP de mesmo tamanho e então repetido para cada pixel na imagem. O LBP original opera com uma vizinhança fixa e tamanho de 3x3 *pixels*.

Assim, é possível construir um histograma de 256 blocos de códigos LBP para a construção do nosso vetor final. A representação proposta por Ahonen et. Al (2004) em "*Face Recognition with Local Binary Patterns. Computer Vision*" é dividir a imagem LBP em regiões locais e extrair um histograma de cada uma delas. Dividimos o rosto igualmente em células de 7×7. O valor de 7x7 foi escolhido pois é sugerido por Ojala et al. (2002), no artigo "*Multiresolution Gray-Scale e Rotation Invariant Texture Classification with Local Binary Patterns*". É retirado o LBPH para cada bloco na imagem. O histograma LBP contém informações em três niveis distintos: O código LBP individual contém informações no nível de *pixel*, os histogramas locais contêm informações em um nível regional de *pixels*, e os histogramas regionais concatenados representam a descrição global da imagem. O grupo de 49 Histogramas LBP é então obtido concatenando os histogramas locais em vectores característicos, ao invés de mesclá-los, assim, podem ser utilizados para aprendizagem de máquina para reconhecimento de face.

Figura 9 – Extração dos Histogramas de Padrões Binários Locais



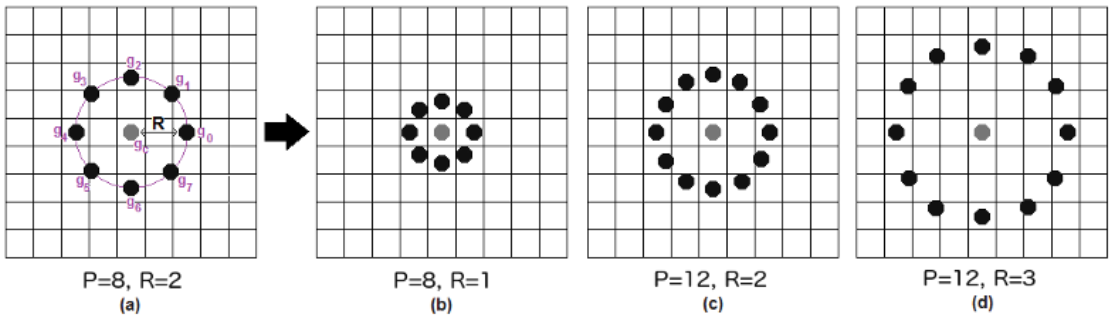
Fonte: Medium (2017).

O benefício principal desta implementação original de LBP é a possibilidade de capturar detalhes extremamente granulares na imagem. No entanto, ser capaz de capturar detalhes em uma escala tão pequena é também a maior desvantagem do algoritmo, pois não é possível capturar detalhes em escalas variáveis, apenas a escala fixa de 3x3. Por esse motivo o procedimento LBP foi expandido para usar um número diferente de raio e vizinhos, que é chamado Circular LBP. Isso pode ser feito

usando interpolação bi linear. Se algum dos pontos de dados estiverem entre os *pixels*, ele usa os valores dos 4 *pixels* mais próximos (2x2) para estimar o valor do novo ponto de dados. Para isso, dois parâmetros foram introduzidos:

- O número de pontos p em uma vizinhança circularmente simétrica (isso permite que não dependa de uma vizinhança quadrada).
- O raio do círculo r, que nos permite contabilizar diferentes escalas.

Figura 10 – Modelo Circular LBP utilizado para escalas variáveis de tamanho



Fonte: Towards Data Science (2017).

Essas etapas são feitas automaticamente no módulo FaceRecognizer do OpenCV, quando uma face é detectada. Esses dados são salvos em um arquivo de extensão (.yml) que fica disponível para o acesso ao recurso de reconhecimento facial.

Após as imagens serem treinadas com o algoritmo, os histogramas criados são usados para representar cada imagem do conjunto de dados do treinamento, que foi salvo no arquivo “trainer.yml”. Assim, o reconhecimento facial em tempo real já pode ser feito criando um histograma para a imagem de entrada para que possa ser feita a comparação com os histogramas salvos no treinamento e retornar o histograma mais próximo. Um exemplo para se comparar o histograma da imagem de entrada com os do treinamento salvo é através da distância euclidiana.

Figura 11 – Fórmula para medir distância euclidiana entre os histogramas

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

Fonte: Towards Data Science (2017).

Após o algoritmo comparar os histogramas do treinamento com o histograma de entrada, ele retorna dois dados:

- O ID correspondente ao histograma de treinamento mais parecido com o histograma de entrada, assim pode-se obter o nome do usuário corresponde ao ID que foi criado utilizando a biblioteca Pickle.
- O algoritmo retorna um número equivalente a diferença entre os histogramas, podendo assim ser usada como uma medida de "confiança". Quanto menor for esse número é melhor, pois significa que os histogramas comparados são mais parecidos.

2.5 Controle de Dispositivos

Para tornar possível o controle de dispositivos, foi desenvolvido um módulo especifico para isso dentro do *website*, onde foi elaborada com uma UX (U*ser Interface*) com a finalidade de ser intuitiva, limpar e a responsividade do *website* foi primordial, para que assim tornasse possível o acesso através de qualquer dispositivo que possua um *browser* de internet (celular, tablet, computador, notebook etc.). O comando dado pelo usuário através da UX é recebido como uma entrada em um código construído em JavaScript, o evento é interpretado e o comando é enviado via JSON[1] para o *software* desenvolvido em Python, responsável por efetuar a comutação com o relé, enviando um sinal alto ou baixo (ligado ou desligado).

[1] *JavaScript Object Notation* ou Notação de Objetos JavaScript (JSON) é um formato de troca de dados entre sistemas independente de linguagem de programação derivado do JavaScript.

3 MODELAGEM

Neste capítulo é feita a documentação do sistema, que inclui diagramas de caso de uso e de sequência, ambos desenvolvidos seguindo a Linguagem Unificada de Modelagem (UML[2] - *Unified Modeling Language*) de forma que facilite a compreensão do projeto.

3.1 Casos De Uso

Os diagramas de caso de uso descrevem um cenário de funcionalidades do ponto de vista do usuário, catalogando os requisitos funcionais do sistema. Dentro do diagrama são retratados os atores (representado pelos bonecos), as funcionalidades (representadas pelos balões com a ação escrita por dentro) e as relações (representadas pelas linhas). Os atores que interagem com o sistema são: Usuário, Administrador e a Interface de programação de aplicativos (API[3] - *Application Programming Interface*) do Plotly:

- **Usuário** é o ator que representa os utilizadores deste aplicativo. Um ator pode, por exemplo, ser reconhecido pelo sistema de reconhecimento facial.
- **Administrador** representa o ator que tem total controle do sistema, podendo acessar os dados de controle, ligar e desligar dispositivos e cadastrar usuários para o *dataset* do reconhecimento facial.
- **API do Plotly** representa o ator da API que permite a interação entre o sistema e o *website* do Plotly, por exemplo, enviando os dados de temperatura e umidade para a plataforma do Plotly, que poderão ser analisados de diferentes formas e exportados.

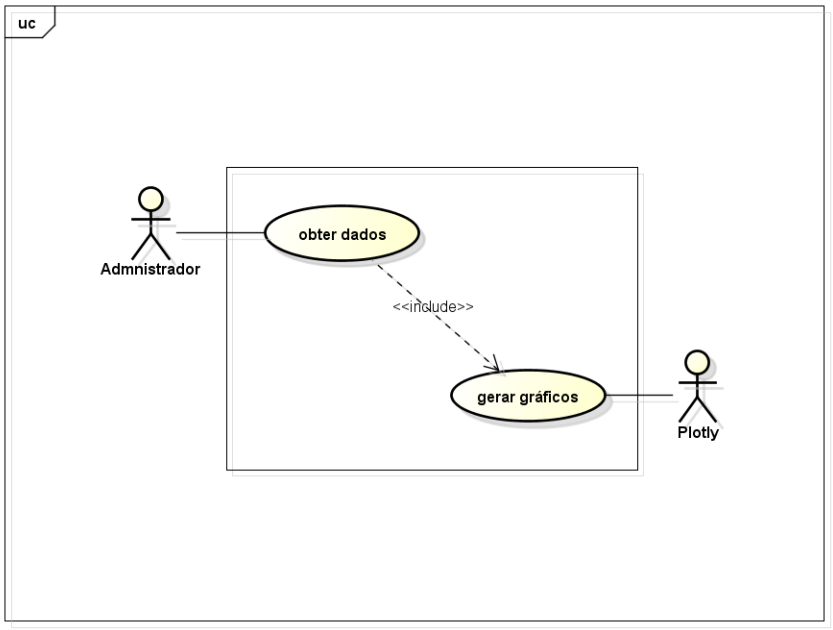
A Figura 12 apresenta o caso de uso para a entrada do usuário no sistema:

---

[2] *Unified Modeling Language* ou Linguagem Unificada de Modelagem (UML) é uma linguagem padrão para modelagem e documentar os sistemas orientados a objetos.

[3] *Application Programming Interface* ou Interface de programação de aplicativos (API) é um conjunto de rotinas, protocolos e ferramentas para a construção de aplicações de software.

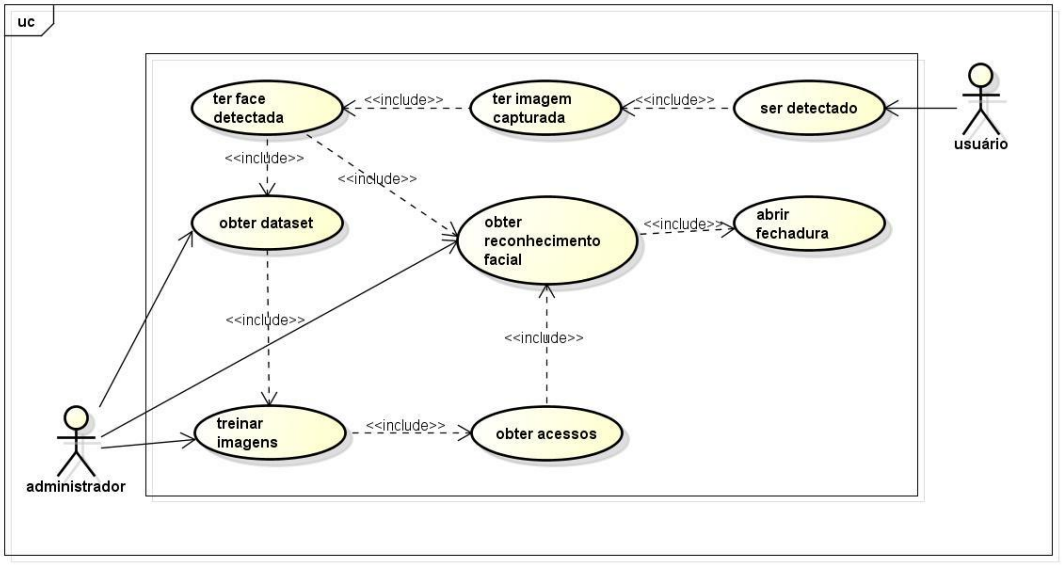
Figura 12 – Diagrama de caso de uso de gráficos do Plotly



Fonte: Elaborado pelo autor.

A Figura 13 apresenta o caso de uso para a entrada o reconhecimento facial:

Figura 13 – Diagrama de caso de uso do reconhecimento facial



Fonte: Elaborado pelo autor.

No subcapítulo 3.2 será apresentada a documentação dos casos de uso deste trabalho.

3.2 Documentação dos Casos de Uso

Cada funcionalidade dos diagramas de casos de uso será descrita com mais detalhes na Tabela 4 e 5.

Tabela 4 – Caso de uso "Gráficos do Plotly"

Nome do caso de uso	Gráficos do Plotly
Atores envolvidos	Administrador e API Plotly
Objetivo	Este caso de uso descreve os passos de envio de dados do sistema para obtenção de gráficos do Plotly
Prioridade de desenvolvimento	Média
Ações do ator	**Ações do Sistema**
1. O administrador clica no botão com o nome "Plolty"	
	2. O sistema leva os dados para a API do Plotly que se encarrega do processo de geração dos gráficos e retorna com as informações.
	3. Ao retornar com as informações, o Plotly envia um *link* de acesso ao gráfico.
4. Após o envio do link, o administrador pode abrir o link que é aberto em um navegador de internet.	
Validações	

Fonte: Elaborado pelo autor.

Tabela 5 – Caso de uso "Reconhecimento Facial"

Nome do caso de uso	Reconhecimento Facial
Atores envolvidos	Usuário e Administrador
Objetivo	Este caso de uso descreve os passos do processo de reconhecimento facial incluindo todas as etapas até a abertura da fechadura.
Prioridade de desenvolvimento	Alta
Ações do Administrador	**Ações do Usuário**
	1. O usuário é detectado pelo sensor de movimento.

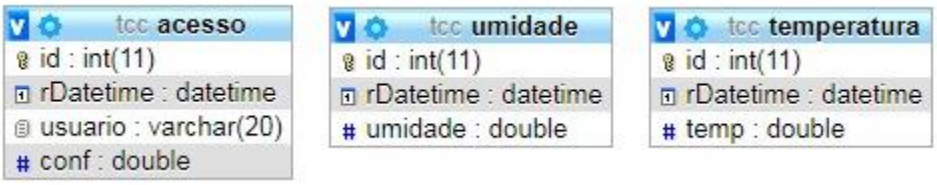
2. O Kinect inicia a captura de vídeo.	
3. Detecta a face que aparece no vídeo.	
4. Tira fotos da face em escala de cinza e armazena no *dataset*.	
5. Treinamento das imagens com o algoritmo de reconhecimento facial.	
6. Algoritmo retorna os vetores de características do usuário e armazena para gerar os acessos.	
7. O algoritmo reconhece a face com base na imagem de entrada e no treinamento salvo.	
8. A permissão de acesso é concedida e a fechadura é aberta.	
	9. O usuário tem a permissão de acesso ao ambiente restrito.
Validações	Armazenamento da imagem no *dataset* apenas se reconhecer um rosto.

Fonte: Elaborado pelo autor.

3.4 Banco de Dados

A base de dados foi criada utilizando o SQlite 3, possui três tabelas, elas não possuem relacionamento pois são utilizadas apenas para o log dos dados de acesso, temperatura e umidade.

Figura 14 – Estrutura das tabelas no banco de dados

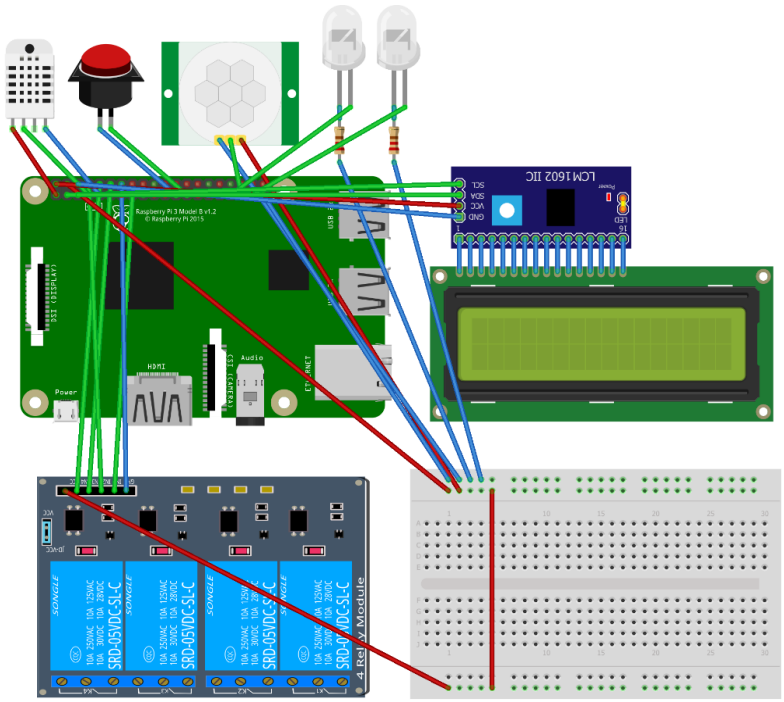


Fonte: Elaborado pelo autor.

3.3 Eletrônica

O esquema a seguir representa o desenvolvimento da eletrônica do projeto, nele pode ser visto como foi feita a ligação dos componentes e sensores através das portas GPIO do Raspberry Pi. Foram utilizados o módulo relé de 4 canais para o controle de dispositivos incluindo a fechadura elétrica, leds com resistores para limitar a corrente e coloca-los no painel do *hardware* que foi montado, display LCD 16x2 para a exibição de informações, botão para acionamento da fechadura elétrica, sensor de presença para que somente execute o sistema de reconhecimento facial quando detectar movimento e o sensor de temperatura e umidade AM-2302, para obter dados de temperatura e umidade do ambiente.

Figura 15 – Esquema de Ligação dos Componentes ao Raspberry Pi



Fonte: Elaborado pelo autor.

3.3.1 Sensor de Temperatura e Umidade AMT-2302

O AMT-2302 é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre -40 a +80 graus Celsius e umidade entre 0 a 100%. Ele é formado por um sensor de umidade capacitivo e um termístor para medir o ar ao redor, enviando ao pino de dados um sinal digital calibrado de alta precisão. A

tecnologia embarcada nos sensores assegura sua confiabilidade e estabilidade. Cada sensor deste modelo é compensado e calibrado em câmara de calibração precisa e o coeficiente de calibração é salvo na memória. Tem tamanho pequeno, baixo consumo de energia, suporta longa distância de transmissão (100m) e conexão com apenas 3 pinos o que torna a conexão muito conveniente.

3.3.2 Módulo Relé de 4 canais

De acordo com o fabricante cada canal precisa de uma corrente de 15-20mA. O relé pode ser utilizado para controlar vários aparelhos e equipamentos desde que não ultrapasse 10 ampères por canal. Ele é equipado com relés que funcionam com até 250 volts em corrente alternada a 10 ampères ou até 30 volts em corrente continua a 10 ampères. Tem uma *interface* padrão que pode ser controlada diretamente pelo microcontrolador. Os princípios eletrônicos do relé são os seguintes: Quando a porta de sinal estiver em nível baixo, a luz de sinalização acenderá e o acoplador óptico 817c, que transforma sinais elétricos por luz e pode isolar sinais elétricos de entrada e saída, conduzirá, e então o transistor conduzirá energia, fazendo com que a bobina do relé seja eletrificada e o contato normalmente aberto do relé será fechado. Quando a porta de sinal estiver em nível alto, o contato normalmente fechado do relé, será aberto. Assim, é possível conectar e desconectar a carga controlando o nível da porta do sinal de controle. Utilizando as portas GPIO do Raspberry Pi, foi instalado um relé de 4 canais, que é utilizado para controle de dispositivos e da fechadura eletrônica utilizada no modulo de controle de acesso por reconhecimento facial. O módulo relé de 4 canais possui saídas digitas que pode-se, através do relé, controlar cargas maiores e dispositivos como motores AC/DC, eletroímãs, solenóides, lâmpadas incandescentes e eletrodomésticos, por exemplo. O modulo relé permite a integração com uma ampla gama de microcontroladores como Arduino, AVR, PIC, ARM e com Raspberry Pi, como foi utilizado neste projeto, através da biblioteca RPi.GPIO, que é utilizada para controlar uma variedade de dispositivos conectados a portas GPIO.

3.3.3 Display LCD 16x2

Displays de LCD podem ser usados para diversas aplicações, como terminais de pontos de venda, instrumentos, computadores, enfim, qualquer coisa que exija uma visualização facilitada dos dados. O display é composto com uma matriz de ponto alfanumérica inteligente de 16 x 2. A tela é capaz de exibir 224 caracteres e símbolos diferentes (podendo ocorrer variações dependendo da fabricante do LCD). O display é conectado utilizando um microcontrolador com conexão i2c, onde torna possível a ligação do display ao Raspberry Pi utilizando apenas quatro fios, sendo eles os pinos SDA, SCL, GROUND e VCC, com a tensão de 5v, economizando portas GPIO do Raspberry Pi e deixando portas livres para o desenvolvimento do projeto.

Figura 16 – Soldagem do modulo i2c ao Display LCD 16x2

Fonte: Elaborado pelo autor.

3.3.4 Criação de extensão com Tomadas Independentes

Para que houvesse uma instalação simplificada e limpa dos dispositivos aos relés, foi criada uma extensão de tomadas com circuitos independentes. Foi efetuada a solda do condutor de cada canal separadamente para que quando a extensão é ligada à energia, uma fase a alimenta diretamente através do barramento que é interligado a todas as tomadas e a outra vai para canal do relé, que retorna para cada tomada separadamente, funcionando como um interruptor. Assim, os

dispositivos são ligados à extensão e podem ser ligados e desligados de forma independente através dos quatro canais do relé.

Figura 17 – Elaboração de extensão de tomadas com circuitos independentes

Fonte: Elaborado pelo autor.

3.3.5 Desenvolvimento da caixa metálica e montagem dos componentes

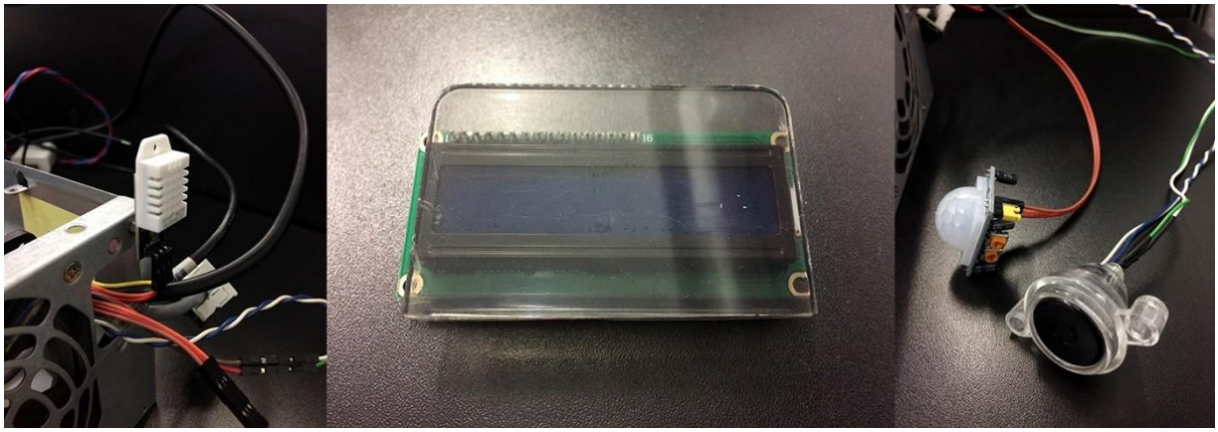
Neste projeto foi construída uma caixa metálica para acomodar o Raspberry Pi, relés, sensores e toda a parte eletrônica do projeto de forma que tudo se mantenha compacto, organizado e seguro. Na caixa metálica foi necessário efetuar um recorte do metal na lateral da caixa no tamanho de 7x2 cm utilizando uma esmerilhadeira, e as rebarbas do metal foram retiradas com o auxílio de uma lima, assim, dando o acabamento ao recorte, finalizando com os furos para a fixação do display e colocação dos LEDs utilizando uma furadeira. A base que serve de sustentação para os componentes foi feita utilizando um plástico grosso que foi recortado no tamanho correto da caixa metálica, que posteriormente foi fixado por parafusos. Foi necessário criar roscas no plástico para que os relés pudessem ser fixados através de espaçadores com rosca, a montagem pode ser vista na Figura 18:

Figura 18 – Montagem dos componentes e parte eletrônica na caixa metálica

Fonte: Elaborado pelo autor.

O sensor de temperatura e umidade foi fixado ao lado da caixa metálica e na posição vertical para que a leitura do sensor seja feita da melhor forma possível. Foi colocada uma peça de acrílico entre a caixa metálica e o *display*, antes da fixação do *display* LCD no local recortado, para que o proteja e proporcione um bom acabamento. Foi instalada um protoboard com dois barramentos, para que fosse feita uma melhor distribuição da alimentação de 5v e GROUND. Foi instalada uma ventoinha 12 volts, alimentada por uma fonte externa que também é utilizada para a abertura da fechadura elétrica. A ventoinha foi instalada para que se obtenha o equilíbrio da temperatura dos componentes internos, aumentando a vida útil dos deles. Ao lado da ventoinha existe uma abertura por onde passam os cabos HDMI, rede, do botão para abertura da fechadura e do sensor PIR, como pode ser visto na Figura 19:

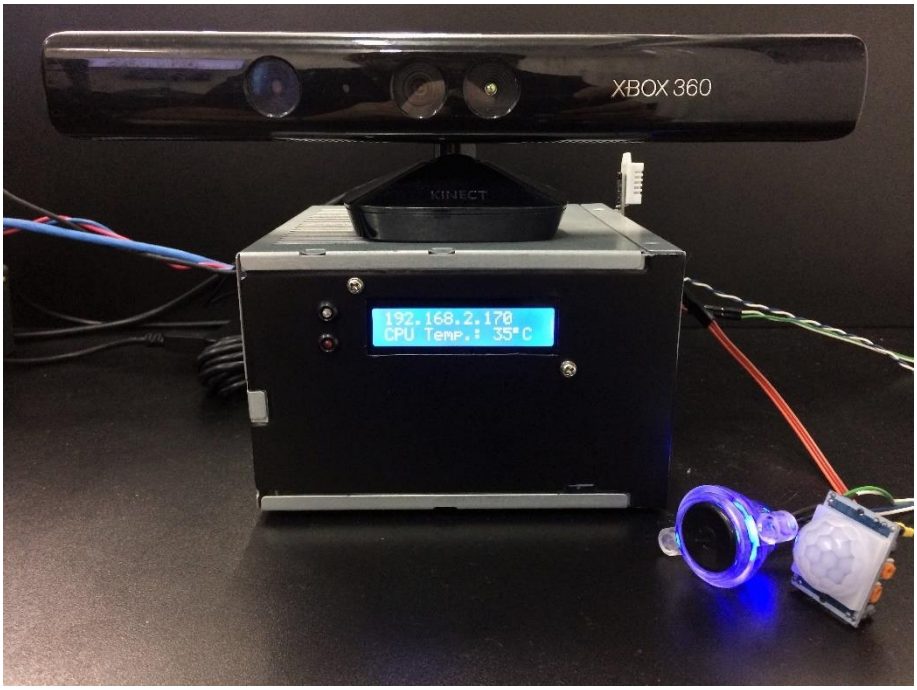
Figura 19 – Componentes eletrônicos do projeto



Fonte: Elaborado pelo autor.

Com todos os componentes montados e organizados de forma segura, foi feito o acabamento da parte metálica com uma película na cor preta, dando o acabamento ao projeto, como pode ser visto na Figura 20:

Figura 20 – Resultado da montagem dos dispositivos

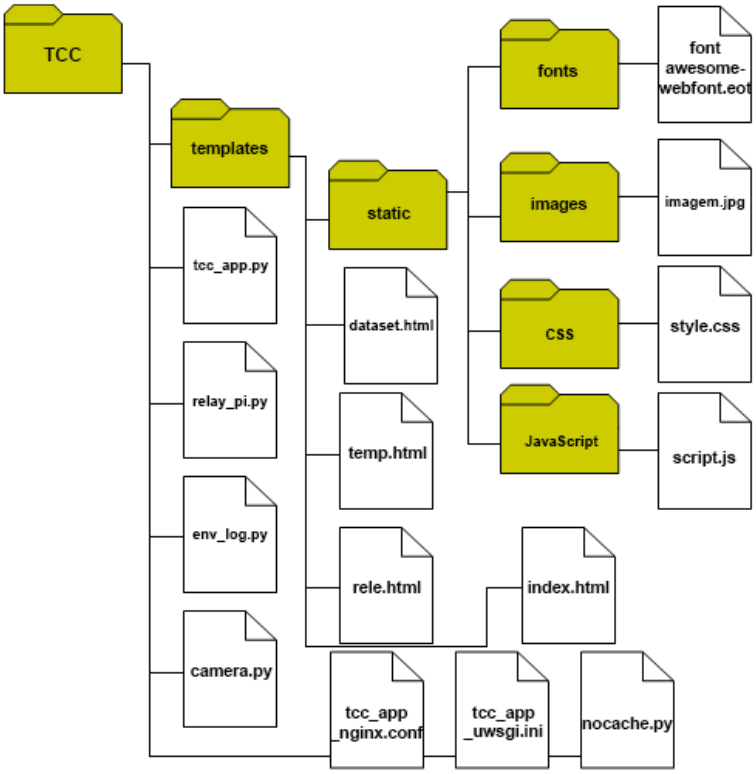


Fonte: Elaborado pelo autor.

4 DESENVOLVIMENTO

Neste capítulo é abordado o desenvolvimento das aplicações integradas a este projeto, contendo módulos *desktop* e *web*. O módulo web possuí *interface* responsiva que se adapta a largura de tela de diferentes dispositivos.

Figura 21 – Estrutura de arquivos do website



Fonte: Elaborado pelo autor.

O módulo web dispõe de diversos recursos, como:

- Controle de dispositivos de corrente alta através de relés.
- Controle e visualização de temperatura e umidade através da leitura e armazenamento dos dados do sensor AM-2302.
- Contém gráficos e tabelas que podem ter seus dados analisados, cruzados e monitorados em tempo real.
- Monitoramento de vídeo através de *stream* de vídeo da webcam.
- Dados de acesso de usuários do sistema de reconhecimento facial.

O módulo *desktop* é composto basicamente por um algoritmo Haar Cascade que foi previamente treinado por Lienhart et. Al. (2002) para detecção frontal de faces e três algoritmos principais que executam as funções essenciais para o reconhecimento facial.

Figura 22 – Estrutura de arquivos do programa de reconhecimento facial

Fonte: Elaborado pelo autor.

- O algoritmo dataset.py captura as imagens de rosto do usuário e armazena na pasta de imagens.
- O algoritmo trainer.py é responsável por treinar as imagens gerando vetores característicos que serão comparados com os da imagem de entrada.
- O arquivo trainer.yml armazena o conjunto de dados resultante do treinamento das imagens do dataset.
- O algoritmo rf.py é responsável pela conversão dos dados de entrada em vetores característicos e comparação com os dados treinados.

- O algoritmo haarcascade_frontalfaces_default.xml é um algoritmo previamente treinado para a detecção frontal de faces.
- O arquivo labels.pickle armazena de forma binária as etiquetas com o id e nome dos usuários.
- O algoritmo colors.py foi desenvolvido para que a interface seja mais amigável e intuitiva, pois o sistema foi desenvolvido de forma estruturada sendo executado por uma janela de terminal.

A linguagem Python foi a mais utilizada neste projeto, estando presente tanto no módulo *web* quanto no módulo *desktop*. O desenvolvimento web com Python foi possível através do serviço WSGI contido no micro *framework* Flask. As bibliotecas utilizadas neste projeto foram estudadas quanto a utilização de recursos, sendo escolhidas as mais eficientes, para que assim, possam ter um bom funcionamento quando instaladas no Raspberry Pi, que apesar de possuir um processador de quatro núcleos e diversos recursos interessantes, possui recursos limitados, como a memória RAM, por exemplo. As bibliotecas utilizadas contêm as mais variadas tecnologias, entre elas então: Flask, OpenCV 3, OpenKinect, Plotly, SQlite3, Text to Speech, entre outras.

4.1 Reconhecimento de Faces

Para a aplicação do reconhecimento facial, são necessários que diversos procedimentos sejam completos, extraindo as imagens de faces em escala de cinza e armazenando-as, treinando as imagens para a extração do LBPH, a partir de então, o algoritmo de reconhecimento facial é capaz de comparar os dados obtidos com base no treinamento das imagens dos usuários com a imagem de entrada da câmera RGB do Kinect.

4.1.1 O comando detectMultiScale

O comando chave do reconhecimento facial que aplica os modelos já implementados no OpenCV às imagens recebe os seguintes argumentos: a própria imagem, um fator escalar que pode ser corrigido conforme o tamanho da imagem para melhorar a qualidade da classificação e o tamanho mínimo da área a ser investigada pelo classificador em *pixels*. Sendo assim, áreas que forem menores do que o estabelecido serão ignoradas.

Sintaxe do código:

```
detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) -> objects
```

Parâmetros:

- image - Imagem onde deve ser detectado o objeto.
- scaleFactor - Ajuste de escala da imagem
- minNeighbors - Mínimo de vizinhos que o objeto deve ter para que ele seja armazenado.
- minSize - Tamanho mínimo do objeto. Objetos menores serão descartados.
- maxSize - Tamanho máximo do objeto. Objetos maiores serão descartados
- objetcs - Vetor de retângulos onde serão guardados os objetos detectados.

4.1.2 Classe LBPHFaceRecognizer

A classe LBPHFaceRecognizer é a responsável por criar os histogramas de padrões binários locais que são obtidos através de análises de textura da imagem, criando vetores característicos que podem ser utilizados para reconhecer uma face.

Sintaxe do código:

```
cv2.face.LBPHFaceRecognizer_create ( int radius = 1, int neighbors = 8,int grid_x = 8, int grid_y = 8, double threshold = DBL_MAX)
```

A classe para utilização do histograma de padrões binários locais (LBPH) utiliza 5 parâmetros:

- Radius (raio): O raio é usado para construir o padrão binário local circular e representa o raio ao redor do *pixel* central. Geralmente é definido como 1.
- Neighbors (vizinhos): O número de pontos de amostra para construir o padrão binário local circular. Lembre-se: quanto mais pontos de amostra você incluir, maior será o custo computacional. Geralmente é definido como 8.
- Grid X: O número de células na direção horizontal. Quanto mais células, quanto mais fina a grade, maior a dimensionalidade do vetor de recursos resultante. Geralmente é definido como 8.
- Grid Y: O número de células na direção vertical. Quanto mais células, quanto mais fina a grade, maior a dimensionalidade do vetor de recursos resultante. Geralmente é definido como 8.
- Limite: O limite aplicado na previsão. Se a distância até o vizinho mais próximo for maior que o limite, esse método retornará -1.

4.1.3 Implementação do Algoritmo de Conjunto de Dados

Neste subcapitulo é apresentado um trecho do código de criação do conjunto de dados de faces do usuário, onde é possível ver a implementação, que basicamente segue esta estrutura:

- Inicia o recebimento da imagem de entrada com o Kinect.
- A imagem é convertida em escala de cinza.

- O comando detectMultiscale recebe os parâmetros para o ajuste da detecção.
- Inicia a repetição onde a cada detecção de face um quadro é armazenado, contendo somente o rosto, para a criação do conjunto de dados do usuário.
- O programa exibe uma mensagem de "*Dataset* criado!" e é encerrado após completar 400 imagens de faces.

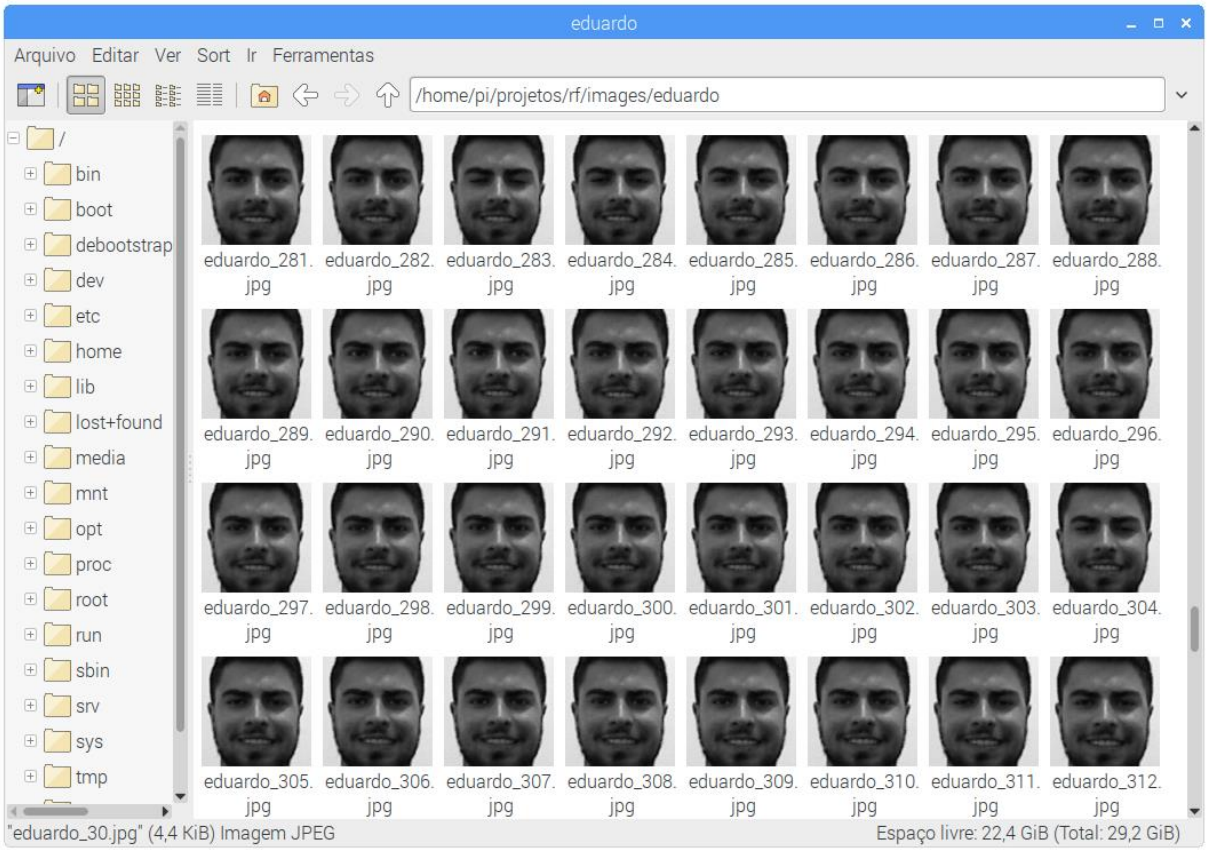
Figura 23 –Trecho do algoritmo de criação do conjunto de dados de imagens

```
while(True):
    image_frame,_ = freenect.sync_get_video()                                   # recebe video do Kinect
    gray = cv2.cvtColor(image_frame,cv2.COLOR_BGR2GRAY)                         # converte imagem em escala de cinza
    faces = faceCascade.detectMultiScale(
                                        gray,
                                        scaleFactor = 1.3,
                                        minNeighbors = 10,
                                        minSize = (30,30),
                                        flags = cv2.CASCADE_SCALE_IMAGE
                                    )                                           # ajustes dos parametros de detecção

    for(x,y,w,h) in faces:
        cv2.rectangle(image_frame, (x,y), (x+w, y+h), (0, 206, 201), 2)         # cria um retangulo em volta da face
        count += 1
        cv2.imwrite('images/' + user + '/' + user + '_' + str(count)+'.jpg', gray[y:y+h, x:x+w]) # salva imagens com o nome do usuário
        frames = cv2.resize(image_frame, (1280, 720), interpolation = cv2.INTER_CUBIC)          # redimensiona imagem de entrada
        cv2.imshow("Criando Dataset",frames[::1,::1,::-1])                      # exibe imagem do Kinect

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break                                                                   # pressione "q" e o programa é interrompido

    elif count >= 400:                                                          # define o numero de fotos tiradas
        print(bcolors.CGREEN2 + bcolors.CBOLD + 'Dataset criado!' + bcolors.CEND)   # exibe mensagem quando terminar
        break                                                                   # finaliza o programa

# Fecha a janela
freenect.sync_stop()
cv2.destroyAllWindows()
```

Fonte: Elaborado pelo autor.

O resultado do algoritmo de criação do conjunto de dados de faces pode ser visto na Figura 24:

Figura 24 – Conjunto de imagens contendo faces em escala de cinza



Fonte: Elaborado pelo autor.

4.1.4 Implementação do Algoritmo de Treinamento

Neste subcapitulo é apresentado um trecho do código do algoritmo de treinamento, que as imagens são treinadas gerando uma saída de vetores armazenados no arquivo “trainer.yml”, a estrutura básica do algoritmo segue a ordem a seguir:

- Captura as imagens dos usuários nas pastas.
- Detecta as faces nas imagens utilizando as propriedades do arquivo “haarcascade_frontalface_default.xml”, que é previamente treinado para detectar rostos.
- Após a detecção o comando LBPHFaceRecognizer_create analisa as imagens gerando uma saída de vetores.
- Gerar etiquetas correspondentes aos histogramas de padrões binários locais calculados.

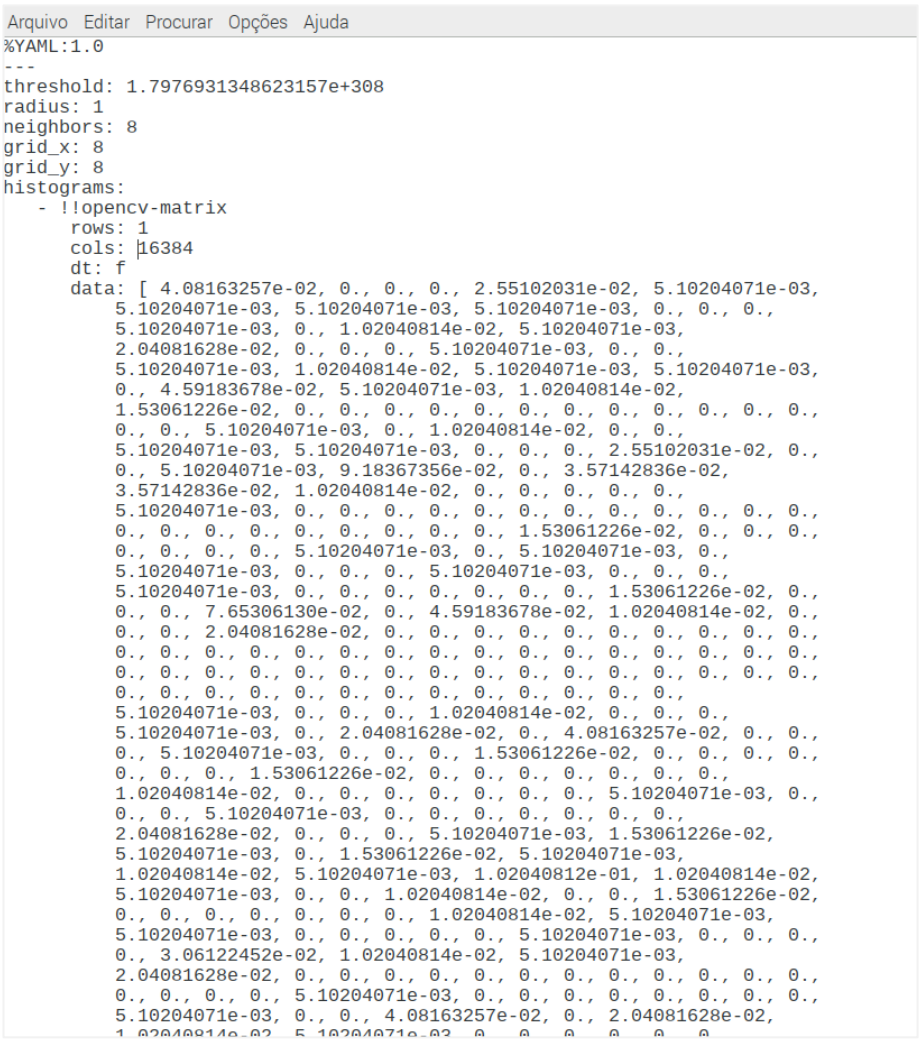
Figura 25 –Trecho do algoritmo de treinamento de imagens de faces

```
print(bcolors.CYELLOW2 + bcolors.CBOLD + 'Treinando imagens...' + bcolors.CEND)
for root, dirs, archivos in os.walk(image_dir):
    for archivo in archivos:
        if archivo.endswith("png") or archivo.endswith("jpg"):
            pathImagem = os.path.join(root,archivo)
            etiqueta = os.path.basename(root).replace(" ", "-").lower()
            # criando as etiquetas
            if not etiqueta in etiquetas_id:
                etiquetas_id[etiqueta] = current_id
                current_id += 1
            id_ = etiquetas_id[etiqueta]
            pil_image = Image.open(pathImagem).convert("L")
            tamanho = (550,550)
            imagenFinal = pil_image.resize(tamanho, Image.ANTIALIAS)
            image_array = np.array(pil_image,"uint8")
            faces = faceCascade.detectMultiScale(image_array,scaleFactor = 1.3,minNeighbors = 10,minSize = (30,30),flags = cv2.CASCADE_SCALE_IMAGE)
            for (x,y,w,h) in faces:
                roi = image_array[y:y+h, x:x+w]
                x_treinamento.append(roi)
                y_etiquetas.append(id_)
with open("labels.pickle",'wb') as f:
    print(bcolors.CYELLOW2 + bcolors.CBOLD + 'Salvando etiquetas...' + bcolors.CEND)
    pickle.dump(etiquetas_id)
print(bcolors.CYELLOW2 + bcolors.CBOLD + 'Salvando treinamento...' + bcolors.CEND)
recognizer.train(x_treinamento, np.array(y_etiquetas))
recognizer.write('trainer.yml')
print('\n' + bcolors.CBLACK + bcolors.CGREENBG +'Sucesso!' + bcolors.CEND)
```

Fonte: Elaborado pelo autor.

Na Figura 26 pode ser visto um trecho do arquivo “trainer.yml” que contém os dados gerados pelo treinamento das imagens:

Figura 26 – Dados gerados pelo algoritmo de treinamento de imagens



Arquivo Editar Procurar Opções Ajuda

```
%YAML:1.0
---
threshold: 1.7976931348623157e+308
radius: 1
neighbors: 8
grid_x: 8
grid_y: 8
histograms:
   - !!opencv-matrix
      rows: 1
      cols: 16384
      dt: f
      data: [ 4.08163257e-02, 0., 0., 0., 2.55102031e-02, 5.10204071e-03,
          5.10204071e-03, 5.10204071e-03, 5.10204071e-03, 0., 0., 0.,
          5.10204071e-03, 0., 1.02040814e-02, 5.10204071e-03,
          2.04081628e-02, 0., 0., 0., 5.10204071e-03, 0., 0.,
          5.10204071e-03, 1.02040814e-02, 5.10204071e-03, 5.10204071e-03,
          0., 4.59183678e-02, 5.10204071e-03, 1.02040814e-02,
          1.53061226e-02, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 5.10204071e-03, 0., 1.02040814e-02, 0., 0.,
          5.10204071e-03, 5.10204071e-03, 0., 0., 0., 2.55102031e-02, 0.,
          0., 5.10204071e-03, 9.18367356e-02, 0., 3.57142836e-02,
          3.57142836e-02, 1.02040814e-02, 0., 0., 0., 0., 0.,
          5.10204071e-03, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 1.53061226e-02, 0., 0.,
          0., 0., 0., 0., 5.10204071e-03, 0., 5.10204071e-03, 0.,
          5.10204071e-03, 0., 0., 0., 5.10204071e-03, 0., 0., 0.,
          5.10204071e-03, 0., 0., 0., 0., 0., 0., 1.53061226e-02, 0.,
          0., 0., 7.65306130e-02, 0., 4.59183678e-02, 1.02040814e-02, 0.,
          0., 0., 2.04081628e-02, 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          5.10204071e-03, 0., 0., 0., 1.02040814e-02, 0., 0., 0.,
          5.10204071e-03, 0., 2.04081628e-02, 0., 4.08163257e-02, 0., 0.,
          0., 5.10204071e-03, 0., 0., 0., 1.53061226e-02, 0., 0., 0.,
          0., 0., 0., 1.53061226e-02, 0., 0., 0., 0., 0., 0.,
          1.02040814e-02, 0., 0., 0., 0., 0., 0., 5.10204071e-03, 0.,
          0., 0., 5.10204071e-03, 0., 0., 0., 0., 0., 0., 0.,
          2.04081628e-02, 0., 0., 0., 5.10204071e-03, 1.53061226e-02,
          5.10204071e-03, 0., 1.53061226e-02, 5.10204071e-03,
          1.02040814e-02, 5.10204071e-03, 1.02040812e-01, 1.02040814e-02,
          5.10204071e-03, 0., 0., 1.02040814e-02, 0., 0., 1.53061226e-02,
          0., 0., 0., 0., 0., 0., 0., 1.02040814e-02, 5.10204071e-03,
          5.10204071e-03, 0., 0., 0., 0., 0., 5.10204071e-03, 0., 0., 0.,
          0., 3.06122452e-02, 1.02040814e-02, 5.10204071e-03,
          2.04081628e-02, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 5.10204071e-03, 0., 0., 0., 0., 0., 0., 0.,
          5.10204071e-03, 0., 0., 4.08163257e-02, 0., 2.04081628e-02,
          1.02040814e-02, 5.10204071e-03, 0., 0., 0., 0., 0., 0.
```

Fonte: Elaborado pelo autor.

4.1.4 Implementação do Algoritmo de Reconhecimento Facial

Neste subcapitulo é apresentado o algoritmo chave do sistema que utiliza os dados de histogramas obtidos no treinamento do conjunto de dados que estão contidos no arquivo “trainer.yml” para a comparação com a entrada de vídeo do Kinect, a fim de verificar se o usuário teve sua face cadastrada no sistema ou não.

Figura 27 –Trecho do algoritmo de reconhecimento facial

```
while True:
    frames,_ = freenect.sync_get_video() # Capturando os frames
    gray = cv2.cvtColor(frames,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,scaleFactor = 1.3,minNeighbors = 10,minSize = (30,30),flags = cv2.CASCADE_SCALE_IMAGE)
    for (x, y, w, h) in faces:
        gray_frames = gray[y:y+h, x:x+w]
        id_, conf = recognition.predict(gray_frames) # Reconhecimento de faces
        if conf >= 4  and conf < 85:                 # definição da confiabilidade
            if conf < 60:                            # quanto menor o número mais preciso
                nome = etiquetas[id_]
                if nome is not None and conf is not None:
                    log_values(nome, conf)
                    print(bcolors.CYELLOW2 + bcolors.CBOLD + "Permissão de acesso concedida para " + nome + "." + bcolors.CEND)
                    print('\n' + bcolors.CBLACK + bcolors.CGREENBG +'Acione o botao para liberar a porta!' + bcolors.CEND + '\n')
                    cv2.putText(frames, nome, (x,y-20), cv2.FONT_HERSHEY_COMPLEX_SMALL , 1, (0, 184, 148), 2, cv2.LINE_AA)
                    try:
                        AudioBotao = gTTS(text=nome + ' , aperte o bot&#227;o para liberar a porta.', lang='pt-br')
                        AudioBotao.save("/tmp/audio.mp3")
                        os.system("mpg321 /tmp/audio.mp3")
                    except:
                        print(bcolors.CRED2 + "Assistente de voz indisponível." + bcolors.CEND)
                    while ok == False:
                        if GPIO.input(pin_btn) == False:
                            try:
                                AudioPermissao = gTTS(text='Bem&#45;vindo, '+ nome, lang='pt-br') # fala de permissao de acesso
                                AudioPermissao.save("/tmp/audio.mp3")
                                os.system("mpg321 /tmp/audio.mp3")
                            except:
                                print(bcolors.CRED2 + "Assistente de voz indisponível." + bcolors.CEND)
                            GPIO.setup(pin_rele, GPIO.OUT)
                            GPIO.output(pin_rele, False)
                            GPIO.output(pin_rele, True)
                else:
                    log_values('none', 0)
        cv2.rectangle(frames, (x, y), (x+w, y+h), (0, 206, 201), 2) # Desenhando um retangulo ao redor das faces
    frames_display = cv2.resize(frames, (1280, 720), interpolation = cv2.INTER_CUBIC)
    cv2.imshow('Controle de Acesso - Reconhecimento Facial', frames_display[::1,::1,::-1])
```

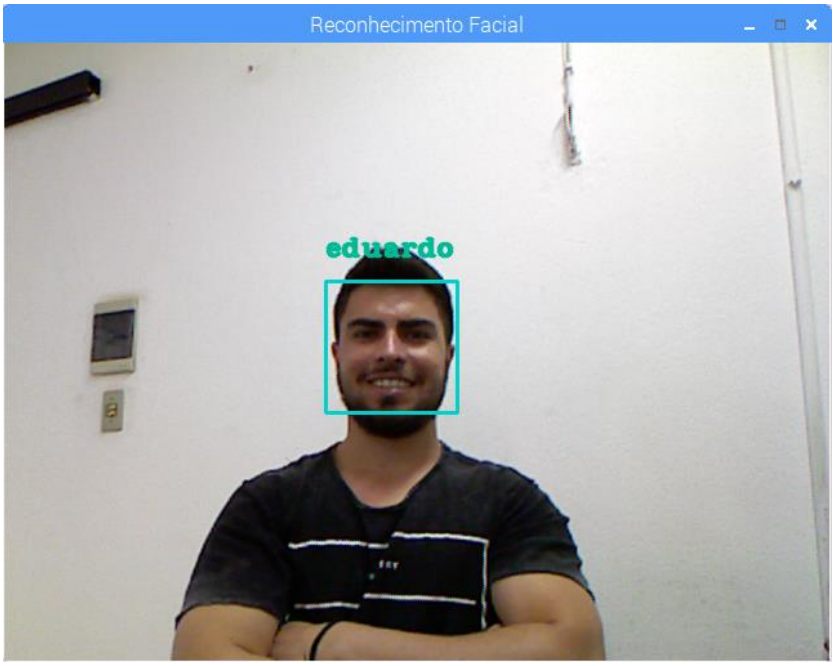
Fonte: Elaborado pelo autor.

A estrutura do algoritmo segue basicamente a ordem a seguir:

- Inicia o recebimento da imagem de entrada com o Kinect.
- A imagem é convertida em escala de cinza.
- O comando detectMultiscale recebe os parâmetros para o ajuste da detecção.
- A classe predict é onde de fato ocorre o reconhecimento facial, atribuindo uma “nota” com base na comparação dos histogramas.
- Se essa “nota” estiver dentro do número de confiabilidade escolhido, o rosto é reconhecido e pelo id do usuário captura-se o nome e exibe na tela.
- Assim, o acesso é liberado e o relé ativado, liberando a fechadura.
- Todo esse processo é feito com a ajuda de uma assistente de voz, que lê as mensagens predefinidas.

Na figura 28 pode ser visto o resultado do algoritmo de reconhecimento facial, que ao detectar uma face desenha um retângulo em volta dela e logo acima é adicionado o nome do usuário que foi reconhecido:

Figura 28 – Reconhecimento facial em tempo real



Fonte: Elaborado pelo autor.

4.2 Instalação do OpenCV 3 e o Python 2.7 no Raspberry Pi

Antes de começar a instalação do OpenCV 3 deve-se ter instalado o sistema operacional Raspbian Stretch no Raspberry Pi, ter acesso físico ao seu Raspberry Pi para que possa abrir um terminal e executar comandos ou acesso remoto via SSH.

Para ter acesso SSH é necessário iniciar o serviço utilizando o comando:

```
$ sudo service ssh start
```

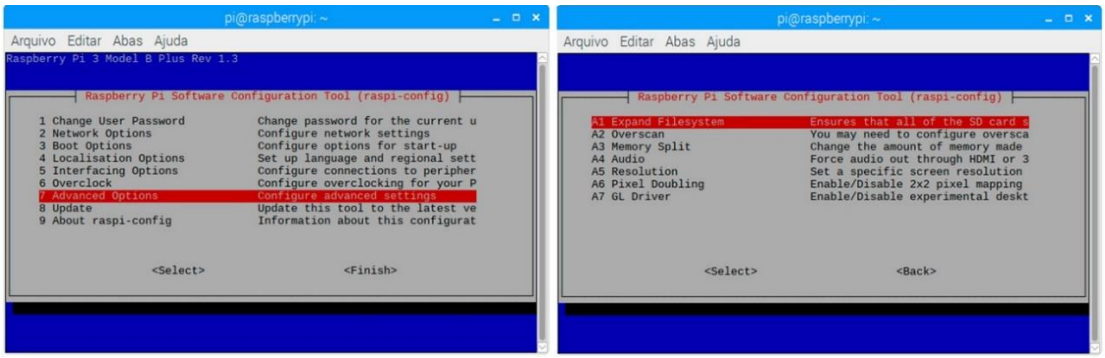
Após feito isso é necessário reiniciar o sistema operacional, deve-se abrir um terminal e digitar o comando para verificar se o serviço ssh foi iniciado corretamente:

```
$ ssh pi @ 127.0.0.1
```

Para expandir o sistema de arquivos, é necessário acessar à página de configurações do Raspberry Pi, através do comando:

```
$ sudo raspi-config
```

Figura 29 – Tela de configuração para expandir o sistema de arquivos



Fonte: Elaborado pelo autor.

Abrirá a interface de configuração de *software* do Raspberry Pi onde será selecionada a opção 7 que corresponde a "Opções Avançadas" em seguida você deve selecionar a primeira opção, “A1. Expanda File System”, pressione Enter no seu teclado, vá para o botão “<Finish>” , e então reinicie o seu Raspberry Pi.
Após a reinicialização o sistema de arquivos deve ter sido expandido para incluir todo espaço disponível em seu cartão micro-SD, pode ser feita essa verificação utilizando o comando "**df -h**" e examinando a saída. Deve-se atualizar o sistema e os pacotes existentes utilizando o comando:

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

Algumas ferramentas de desenvolvedor são necessárias, incluindo o CMake, que é utilizando durante o processo de compilação do OpenCV 3:

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

Instalar pacotes de I/O de imagem e I/O de vídeo para que seja possível a manipulação de diversos formatos de imagem e vídeo:

```
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

Instalar a biblioteca de desenvolvimento do GTK que acompanha um sub modulo chamado **highgui** que pode ser utilizado para criar GUI's básicas:

```
$ sudo apt-get install libgtk2.0-dev
```

Operações de matriz utilizadas pelo OpenCV podem ser otimizadas instalando alguns pacotes importantes para quando se trabalha com dispositivos com recursos limitados como o Raspberry Pi:

```
$ sudo apt-get install libatlas-base-dev gfortran
```

Instalar os pacotes do Python 2.7 e do Python 3 que serão utilizados na compilação do OpenCV onde será feita a ligação e escolha da linguagem de desenvolvimento:

```
$ sudo apt-get install python2.7-dev python3-dev
```

Efetuar o *download* do código-fonte do OpenCV diretamente do repositório oficial do OpenCV no Github, a versão utilizada será a 3.3.0, o arquivo .zip deverá ser descompactado em seguida:

```
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
$ unzip opencv.zip
```

Para ter acesso a recursos como SIFT, SURF e entre outros é necessário efetuar o *download* do repositório adicional do OpenCV para termos uma instalação completa com todos recursos disponíveis:

```
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
$ unzip opencv_contrib.zip
```

Instalar o gerenciador de pacotes Python chamado Pip:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

É tido como uma boa prática utilizar ambientes virtuais para Python, que torna possível manter em locais diferentes dependências utilizadas por projetos distintos, e ainda mantém o diretório **site-packages** limpo e organizado:

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

Após a instalação do virtualenv e virtualenvwrapper, deve-se atualizar o arquivo de perfil utilizando o comando **nano ~/.profile** com as seguintes linhas de código:

```
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

Para carregar o perfil atualizado deve-se efetuar o *login* em outro terminal ou apenas utilizar o comando:

```
$ source ~/.profile
```

Para criar o ambiente virtual Python que será feita a instalação do OpenCV deve ser utilizado o seguinte comando:

```
$ mkvirtualenv cv -p python2
```

Agora é necessário acessar o ambiente virtual e isso pode ser visto examinando sua linha de comando onde o texto **(cv)** precede o mesmo. Para entrar no ambiente virtual criado utiliza-se o comando a seguir:

```
$ workon cv
```

Caso não apareça o (cv) no início do terminal, isto significa que o usuário ainda não entrou no ambiente virtual, para corrigir basta executar os comandos **source** e **workon**, como já foi mencionado acima.

O pacote NumPy é um pacote Python utilizado para processamento numérico e pode ser instalado executando o seguinte comando:

```
$ pip install numpy
```

Antes de instalar o OpenCV é preciso garantir que se encontra no ambiente virtual **(cv)**, caso não esteja basta executar o comando **workon**:

```
$ workon cv
```

Verificando-se que o usuário está dentro do ambiente virtual, deve-se configurar a construção da compilação usando o CMake, porém, é recomendado que seja fechado qualquer programa que esteja aberto no Raspberry Pi, pois este processo exige uma alta quantidade de processamento e pode levar até duas horas:

```
$ cd ~/opencv-3.3.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D BUILD_TESTS=OFF \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D BUILD_EXAMPLES=OFF ..
```

Na instalação serão utilizados os sinalizadores NEON e VFPV3, pois foi verificado que a instalação deles proporciona uma instalação otimizada, com um aumento de desempenho de até 30%, em relação a instalação sem a utilização destes sinalizadores. Para que seja feita uma compilação mais rápida pode-se aumentar o espaço de troca, possibilitando assim que sejam utilizados os quatro núcleos do processador do Raspberry Pi, para isso deve-se editar o parâmetro CONF_SWAPSIZE em /etc/dphys-swapfile utilizando um editor de arquivos:

```
$ nano /etc/dphys-swapfile
```

```
# CONF_SWAPSIZE=100
CONF_SWAPSIZE=1024
```

Reiniciar o serviço de troca:

```
$ sudo /etc/init.d/dphys-swapfile restart
```

Iniciar a compilação utilizando make -j4, onde o número 4 representa a quantidade de núcleos do processador que se deseja utilizar para o processo de compilação:

```
$ make -j4
```

Com a compilação finalizada sem erros, pode-se iniciar a instalação otimizada do Open CV no Raspberry Pi:

```
$ sudo make install
$ sudo ldconfig
```

Após a conclusão da instalação redefina o CONF_SWAPSIZE em /etc/dphys-swapfile para 100MB, este processo exige que o serviço de troca seja reiniciado novamente.

Efetuando a ligação do arquivo cv2.so ao ambiente virtual cv:

```
cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

Para testar se a instalação foi concluída com sucesso antes de começar a desenvolver o desenvolvimento de sistemas de visão computacional, ativar um terminar Python e importar a biblioteca OpenCV:

```
$ source ~/.profile
$ workon cv
$ python
>>> import cv2
>>> cv2.__version__
'3.3.0'
```

OpenCV versão 3.3.0 instalado no Raspberry Pi e pronto para o início do desenvolvimento do sistema de reconhecimento facial.

4.3 Instalação da biblioteca Open Kinect

Primeiramente, é necessário instalar as dependências começando por atualizar os pacotes existentes no Raspbian com os comandos:

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

Instalar dependências para bibliotecas de desenvolvedor:

```
$ sudo apt-get install git-core cmake freeglut3-dev pkg-config build-essential libxmu-dev
libxi-dev libusb-1.0-0-dev
```

Efetuar o *download* do código-fonte do OpenKinect diretamente do repositório oficial do OpenKinect (2018) no Github, a versão utilizada será a do “libfreenect” que é utilizada para controle da primeira versão do Kinect:

```
$ git clone git://github.com/OpenKinect/libfreenect.git
```

Compilar o OpenKinect:

```
$ cd libfreenect
$ mkdir build
$ cd build
$ cmake -L ..
$ make
```

Instalar o OpenKinect:

```
$ sudo make install
$ sudo ldconfig /usr/local/lib64/
```

Para criar regras para o gerenciador de dispositivos do Rasbian Stretch, deve-se abrir o arquivo em /etc/udev/rules.d/51-kinect.rules utilizando um editor de texto:

```
$ sudo nano /etc/udev/rules.d/51-kinect.rules
```

Salvar o conteúdo a seguir no arquivo 51-kinect.rules:

Figura 30 – Regras para o gerenciador de dispositivos

```
1  # ATTR{product}=="Xbox NUI Motor"
2  SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02b0", MODE="0666"
3  # ATTR{product}=="Xbox NUI Audio"
4  SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ad", MODE="0666"
5  # ATTR{product}=="Xbox NUI Camera"
6  SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ae", MODE="0666"
7  # ATTR{product}=="Xbox NUI Motor"
8  SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02c2", MODE="0666"
9  # ATTR{product}=="Xbox NUI Motor"
10 SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02be", MODE="0666"
11 # ATTR{product}=="Xbox NUI Motor"
12 SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02bf", MODE="0666"
```

Fonte: Elaborado pelo autor.

Reiniciar o Raspberry Pi através do comando:

```
$ freenect-glview
```

Uma janela é aberta mostrando a imagem de profundidade da câmera laser e a imagem RGB lado a lado. Pode-se controlar o motor no Kinect pressionando a tecla “w” e “x” no teclado, o que faz com que o Kinect se incline para cima ou para baixo, respectivamente, também é possível centralizar o Kinect automaticamente apertando a tecla “s” no teclado.

Finalmente, a biblioteca Open Kinect é instalada e pronta para ser utilizada para controlar o Kinect no projeto de reconhecimento facial com OpenCV. Para que seja possível utilizar o Kinect juntamente com OpenCV e Python se faz necessária a instalação de *wrappers* Python para a Open Kinect, antes disso é necessário instalar as dependências necessárias:

```
$ sudo apt-get install cython
$ sudo apt-get install python-dev
$ sudo apt-get install python-numpy
```

Acessar o diretório /libfreenect/wrappers/python:

```
$ cd /libfreenect/wrappers/python
```

Executar o script em Python para a instalação dos wrappers:

```
$ sudo python setup.py install
```

Criar um arquivo Python utilizando as bibliotecas OpenCV e OpenKinect. É criado um arquivo Python (.py) nomeando-o, por exemplo, “kinect.py” e inserindo os comandos em Python.

Figura 31 – Obtendo imagem do Kinect utilizando o Open CV 3

```
1
2   import freenect     # importar as funções do OpenKinect
3   import cv2          # importar as funções OpenCV
4   import numpy as np  # necessário para trabalhar com array
5
6   # função para pegar a imagem RGB do Kinect
7   def get_video():
8       array,_ = freenect.sync_get_video()             # array do vídeo RGB
9       array = cv2.cvtColor(array,cv2.COLOR_RGB2BGR)   # define a imagem colorida
10      return array                                    # retorna o vídeo RGB
11
12  # função para pegar a imagem profunda  do Kinect
13  def get_depth():
14      array,_ = freenect.sync_get_depth()             # array da imagem profunda
15      array = array.astype(np.uint8)                  # formatando para uint8
16      return array                                    # retorna imagem profunda
17
18  # função principal do programa
19  if __name__ == "__main__":
20      while 1:
21          frame = get_video()                 # pega os frames da câmera RGB
22          depth = get_depth()                 # pega os frames da câmera profunda
23          cv2.imshow('Imagem RGB',frame)      # mostra imagem RGB
24          cv2.imshow('Imagem Profunda',depth) # mostra imagem profunda
25
26          # o programa se encerra quando "esc" for pressionado
27          k = cv2.waitKey(5) & 0xFF
28          if k == 27:
29              break
30      cv2.destroyAllWindows()  #fecha a janela
```

Fonte: Elaborado pelo autor.

Executar o programa utilizando o comando:

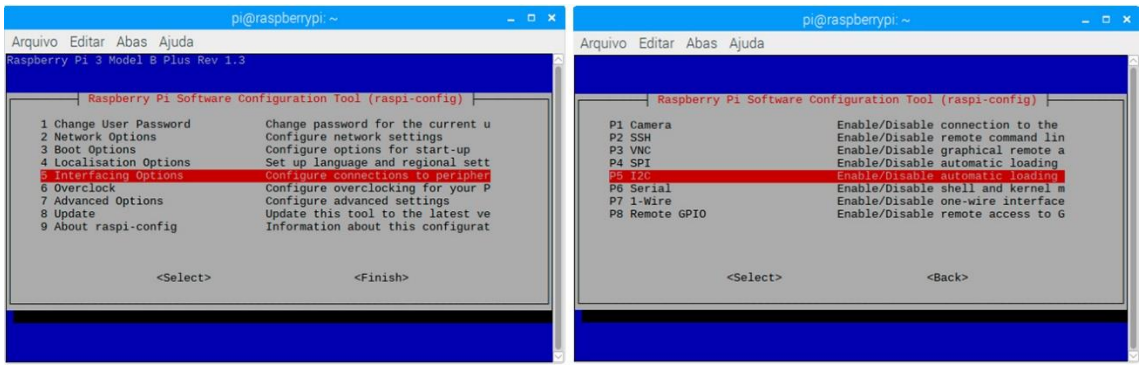
```
$ python kinect.py
```

Retorna duas janelas contendo a imagem profunda e imagem RGB do Kinect. Para encerrar o programa basta pressionar a tecla "esc" no teclado.

4.4 Instalação e configuração do Display LCD 16x2 com o módulo i2c

Para efetuar a configuração do módulo i2c que será usado no display LCD é necessário habilitar a interface I2C no Raspbian abrindo uma janela de terminal e executando o comando **$ sudo raspi-config**, onde será apresentada a tela abaixo:

Figura 32 – Telas de configuração para habilitar interface i2c



Fonte: Próprio autor.

Deve-se adicionar os módulos na inicialização acessando o arquivo em "**/etc/modules**", isso é feito da seguinte forma:

```
$ sudo nano /etc/modules
```

Deve ser inserida duas linhas neste arquivo:

```
i2c-dev
i2c-bcm2708
```

A máquina deve ser reiniciada para que as novas configurações sejam validadas:

```
$ sudo reboot
```

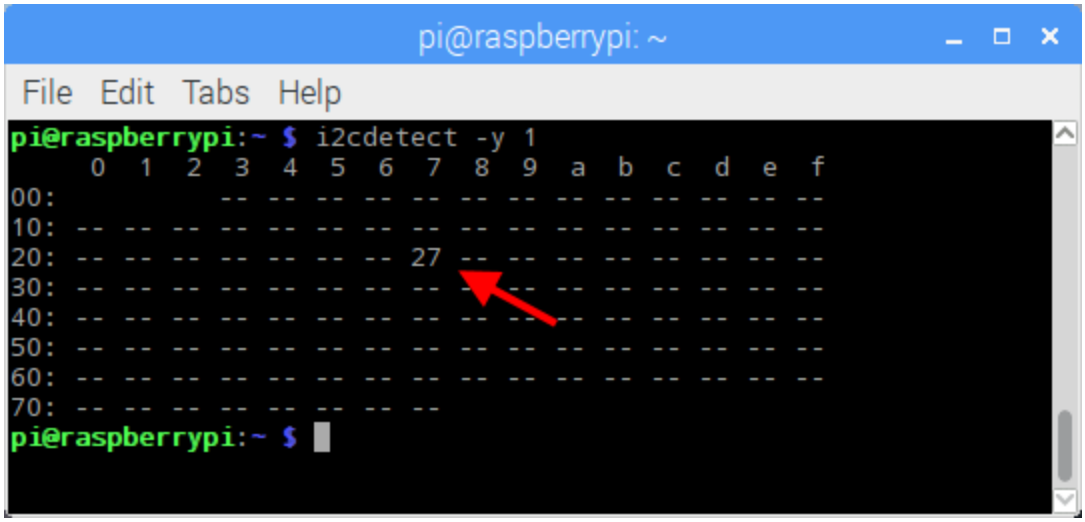
Deve-se instalar as dependências necessárias para o correto funcionamento do display:

```
$ sudo apt-get install python-smbus i2c-tools
```

Utilizar o i2cdetect para ver uma lista dos dispositivos i2c conectados ao Raspberry Pi:

```
$ sudo i2cdetect -y 1
```

Figura 33 – Verificando endereço da interface i2c do Display LDC



Fonte: Elaborado pelo autor.

O endereço do módulo i2c com o *display* é o 27, como pode ser visto na imagem acima. Esse endereço será utilizado para configurar a biblioteca dele.

Deve-se efetuar o *download* do driver em Python onde será possível se conectar com a tela:

```
$ git clone https://github.com/CaptainStouf/raspberry_lcd4x20_I2C
```

```
$ cd ~ /raspberry_lcd4x20_I2C
```

Abrir o arquivo **lcddriver.py** utilizando um editor de texto ou IDE:

```
$ sudo nano lcddriver.py
```

Edite o campo de endereço do LCD caso seja necessário para o endereço que foi obtido através do comando **i2cdetect**, devendo ficar dessa forma:

```
# LCD Address
ADDRESS: 0x27
```

Criando um arquivo de teste para inserir texto no display LCD, deve-se criar um arquivo de teste chamado display_teste.py:

```
$ sudo nano display_teste.py
```

Inserir os códigos abaixo escritos no arquivo Python:

Figura 34 – Exemplo de código que escreve no display LDC

```
1   #  importar as bibliotecas necessárias
2   import lcddriver
3   import time
4
5   # inicalizando o display
6   lcd = lcddriver.lcd()
7
8   # limpando a tela do display
9   lcd.lcd_clear()
10
11  # preenchendo as linhas do display
12  lcd.lcd_display_string("Hello world !", 1)
13  lcd.lcd_display_string("Rasperry Pi", 2)
```

Fonte: Elaborado pelo autor.

Em seguida deve-se executar:

```
$ python display_teste.py
```

Isto deve fazer com que o *display* seja preenchido com o texto escolhido.

O programa desenvolvido em Python fará com que o *display* LCD exiba as seguintes informações:

- Nome do computador (no caso o nome escolhido foi “Servidor”).
- Data e hora atual.
- IP da rede conectada ao Raspberry Pi.
- Temperatura do CPU.
- Temperatura e umidade do ambiente.

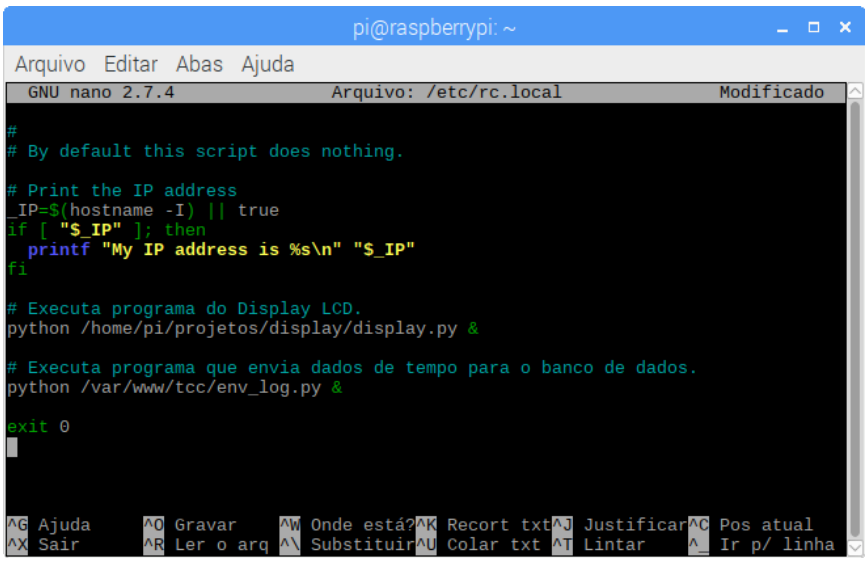
4.5 Executando Programas Automaticamente ao Iniciar o Raspbian

Para executar programas automaticamente ao iniciar o Raspbian é preciso adicioná-lo ao arquivo localizado em "**/etc/rc/local**". O arquivo "**/etc/rc.local**" é um script onde é inserido o que deve ser inicializado logo após o sistema operacional executar todos os serviços. O primeiro passo é acessar o arquivo "**rc.local**" usando:

```
$ sudo nano /etc/rc.local
```

A sintaxe usada é inserir a chamada para o Python, seguido pelo caminho do programa e no final o caractere "**&**", que serve para que o sistema entenda que o programa deverá ser executado em segundo plano. como pode ser visto na imagem a seguir:

Figura 35 – Inicializando Programas Python Automaticamente com o Raspbian



Fonte: Elaborado pelo autor.

4.6 Instalação do Sensor de Umidade e Temperatura AMT-3202.

Efetuar o *download* do código-fonte da biblioteca "Adafruit_DHT" diretamente do repositório:

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

Instalar as dependências:

```
$ sudo apt-get install python-pip
$ sudo python -m pip install --upgrade pip setuptools wheel
```

Instalar o Adafruit_DHT para a leitura dos sensores utilizando a linguagem Python:

```
$ cd ~ /Adafruit_Python_DHT
$ sudo python setup.py install
```

Programa para receber dados de temperatura e umidade do sensor DHT2302 com Python:

Figura 36 – Algoritmo para leitura de umidade e temperatura

```
1  # biblioteca para utilizacao do sensor AM2302
2  import Adafruit_DHT
3
4  # armazena os dados de temperatura e umidade em variaveis
5  humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4)
6      if humidity is not None and temperature is not None:
7          # exibe temperatura e umidade
8          print('Temp.: %d%sC' % (temperature, chr(223)))
9          print('Umidade: %.1f%%' % humidity)
```

Fonte: Elaborado pelo autor.

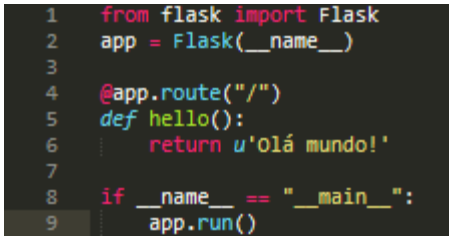
4.7 Instalação do Micro-Framework Flask

Certificando-se de estar dentro do ambiente de desenvolvimento (cv), foi utilizado o seguinte comando para efetuar a instalação:

```
$ pip install Flask
```

Fazer um teste de funcionamento com um "Hello world!", criando um arquivo Python:

Figura 37 – Demonstração da utilização do Flask com Python para Web



```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route("/")
5  def hello():
6      return u'Olá mundo!'
7
8  if __name__ == "__main__":
9      app.run()
```

Fonte: Elaborado pelo autor.

Para iniciar o *script* de código basta fazer a chamada para o Flask:

```
$ FLASK_APP=hello.py flask run
```

O Flask por padrão funciona na porta 5000, podendo ser alterada via código. Quando iniciado o serviço, aparece a seguinte linha no termina:

```
* Running on http://localhost:5000/
```

No entanto, é necessário que o sistema web já esteja inserido no diretório em "/var/www/..." utilizado para o desenvolvimento do website. Este diretório é criado automaticamente pelo servidor web, que deve haver também o arquivo de configuração WSGI do servidor.

4.8 Instalação do Gerenciador de Banco de Dados SQlite 3

Para instalar o SQLite3 deve ser usado o seguinte comando:

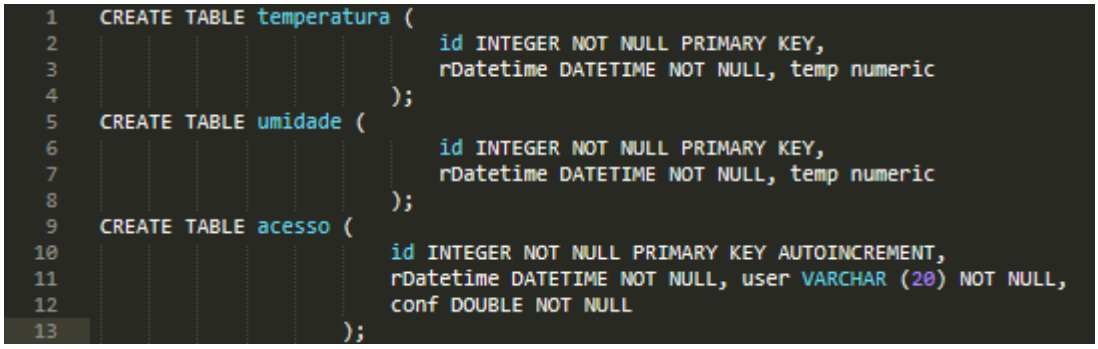
```
$ sudo apt-get instalar sqlite3
```

Com o SQLite 3 instalado no Raspbian do Raspberry Pi. Pode-se criar a base de dados utilizando a chamada para o SQlite 3 em seguida é dado o nome da base de dados, da seguinte forma:

```
$ sqlite3 <nome do arquivo DB>
```

A criação do banco de dados foi feita da seguinte forma:

Figura 38 – Tabelas criadas no banco de dados do SQlite 3



```
CREATE TABLE temperatura (
                            id INTEGER NOT NULL PRIMARY KEY,
                            rDatetime DATETIME NOT NULL, temp numeric
                        );
CREATE TABLE umidade (
                            id INTEGER NOT NULL PRIMARY KEY,
                            rDatetime DATETIME NOT NULL, temp numeric
                        );
CREATE TABLE acesso (
                        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
                        rDatetime DATETIME NOT NULL, user VARCHAR (20) NOT NULL,
                        conf DOUBLE NOT NULL
                    );
```

Fonte: Elaborado pelo autor.

Neste projeto, são utilizadas apenas três tabelas de banco de dados, elas não possuem relacionamento entre si pois são utilizadas apenas para o log dos dados de monitoramento, sendo que a tabela “acesso” armazena os dados de acesso e a tabela “temperatura” e “umidade” armazenam os dados de temperatura e umidade, respectivamente. Esses dados são monitorados através do módulo de controle de acesso e ambiente do *website*.

4.9 Algoritmo de Controle de Relés

O algoritmo de controle de relés, foi desenvolvido utilizando Python 2.7 com uma implementação do Flask para que seja desenvolvido um sistema que funcione via *web*. A biblioteca necessária para o controle dos relés e sensores é a RPi.GPIO, que é a responsável pela configuração das portas GPIO do Raspberry Pi.

Figura 39 – Algoritmo de controle de relés com Python e Flask

```python
relays = []
ports = []
names = []
checkeds = []

@app.route('/static/css/')  # define a rota para a pasta de estilos
@nocache                    # não armazenar o cache de estilos
def static_css(path):
    return app.send_static_file(path)

@app.route('/static/js/')   # define a rota para a pasta de scripts
@nocache                    # não armazenar o cache de scripts
def static_js(path):
    return app.send_static_file(path)

@app.route('/con<string:relay_id>/<string:state>')  # define a rota para o status do rele
@nocache                                            # não armazenar o cache do status do rele
def relay_routing(relay_id,state):
    global relays
    global checkeds
    checkeds[int(relay_id)-1] = str(state)
    relays[int(relay_id)-1].go(state)
    return 'ok'

@app.route("/rele")
@nocache
def rele():
    global ports
    global names
    global checkeds
    return render_template('rele.html',ports=ports,names=names,checkeds=checkeds) # renderiza a página rele.html

def init():
    global ports
    global names
    global relays
    global checkeds

    inverse = True

    ports = ['11','12','13']               # portas GPIO conectadas aos canais dos relés
    names = ['Lampada','Switch','Tomada']  # nome definido para cada canal
    checkeds = ['off' for _ in names]      # status do rele
    relays = [Relay(int(port),inverse) for port in ports
```

Fonte: Elaborado pelo autor.

A interface do botão de ligar e desligar os relés na página *web* é interpretada por um *script* em JavaScript que recebe a requisição via Json, como pode ser visto na figura 27:

Figura 40 – JavaScript para a comunicação do status do relé com o sistema

```javascript
$(document).on('click', 'label', function () {
    click($(this).attr('id'));
    return false;
});

function click(id){
    $("#button" + id).attr("disabled", false);
    if($("#button" + id).is(':checked')){
        $("#button" + id).prop("checked", false);
        execute("/con" + id + "/off")
    } else{
        $("#button" + id).prop("checked", true);
        execute("/con" + id + "/on")
    }
}

function execute(req){
    $.getJSON(req, {}, function(data) {})
}
```

Fonte: Elaborado pelo autor.

4.10 Algoritmo de Controle de Temperatura e Umidade

O algoritmo de controle de temperatura e umidade utiliza a biblioteca Arrow, que é uma biblioteca para formatar datas com facilidade, a utilização dela foi necessária pelo fato de que os filtros de dados disponibilizados no módulo de controle, são filtrados por data, assim, precisam de um tratamento de validação para garantir que as requisições ao banco de dados sejam enviadas corretamente.

Figura 41 – Algoritmo de controle de temperatura e umidade

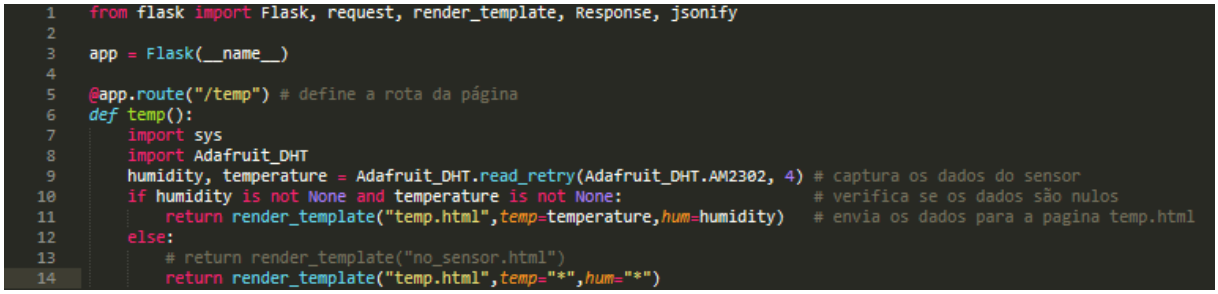
```
def get_records():
    import sqlite3
    from_date_str   = request.args.get('from',time.strftime("%Y-%m-%d 00:00"))  # define o valor como 00:00
    to_date_str     = request.args.get('to',time.strftime("%Y-%m-%d %H:%M"))    # obtem o valor da data do URL
    timezone        = request.args.get('timezone','Etc/UTC');
    range_h_form    = request.args.get('range_h','');
    range_h_int     = "nan"
    try:
        range_h_int = int(range_h_form)
    except:
        print "range_h_form nao e um numero"
    if not validate_date(from_date_str):    # validando a data antes de envia-la ao DB
        from_date_str   = time.strftime("%Y-%m-%d 00:00")
    if not validate_date(to_date_str):
        to_date_str     = time.strftime("%Y-%m-%d %H:%M")
    # Criando o objeto datetime para converter em UTC a partir do horario local do navegador
    from_date_obj       = datetime.datetime.strptime(from_date_str,'%Y-%m-%d %H:%M')
    to_date_obj         = datetime.datetime.strptime(to_date_str,'%Y-%m-%d %H:%M')
    # Se range_h e definido, nao e preciso do filtro de data
    if isinstance(range_h_int,int):
        arrow_time_from = arrow.utcnow().replace(hours=-range_h_int)
        arrow_time_to   = arrow.utcnow()
        from_date_utc   = arrow_time_from.strftime("%Y-%m-%d %H:%M")
        to_date_utc     = arrow_time_to.strftime("%Y-%m-%d %H:%M")
        from_date_str   = arrow_time_from.to(timezone).strftime("%Y-%m-%d %H:%M")
        to_date_str     = arrow_time_to.to(timezone).strftime("%Y-%m-%d %H:%M")
    else:
        from_date_utc   = arrow.get(from_date_obj, timezone).to('Etc/UTC').strftime("%Y-%m-%d %H:%M")   # Converter os dados em UTC...
        to_date_utc     = arrow.get(to_date_obj, timezone).to('Etc/UTC').strftime("%Y-%m-%d %H:%M")     # para recuperar no banco de dados
    conn = sqlite3.connect('/home/pi/projetos/rf/tcc_app.db')
    curs = conn.cursor()
    curs.execute("SELECT * FROM temperatures WHERE rDatetime BETWEEN ? AND ?", (from_date_utc.format('YYYY-MM-DD HH:mm'), to_date_utc.format('YYYY-MM-DD HH:mm')))
    temperatures = curs.fetchall()
    curs.execute("SELECT * FROM humidities WHERE rDatetime BETWEEN ? AND ?", (from_date_utc.format('YYYY-MM-DD HH:mm'), to_date_utc.format('YYYY-MM-DD HH:mm')))
    humidities = curs.fetchall()
    curs.execute("SELECT * FROM acesso WHERE rDatetime BETWEEN ? AND ?", (from_date_utc.format('YYYY-MM-DD HH:mm'), to_date_utc.format('YYYY-MM-DD HH:mm')))
    acesso = curs.fetchall()
    conn.close()
    return [temperatures, humidities, acesso, timezone, from_date_str, to_date_str]
```

Fonte: Elaborado pelo autor.

4.11 Algoritmo de Visualização de Temperatura e Umidade

O sistema de visualização de temperatura e umidade exibe os dados em uma *interface web*, contendo os dados de temperatura e umidade recebidos do sensor AM-2302 com uma imagem de fundo, sendo que os dados de temperatura e umidade são atualizados automaticamente a cada período (10 segundos). Ele foi pensado para ser utilizado como mostrador de temperatura e umidade, sendo utilizado em tela cheia através do navegador de internet, por exemplo, quando o computador está ocioso, ou quando se deseja ter acesso rápido às condições de temperatura e umidade em tempo real.

Figura 42 – Código em Python para a visualização de temperatura e umidade



```python
from flask import Flask, request, render_template, Response, jsonify

app = Flask(__name__)

@app.route("/temp") # define a rota da página
def temp():
    import sys
    import Adafruit_DHT
    humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4) # captura os dados do sensor
    if humidity is not None and temperature is not None:                    # verifica se os dados são nulos
        return render_template("temp.html",temp=temperature,hum=humidity)   # envia os dados para a pagina temp.html
    else:
        # return render_template("no_sensor.html")
        return render_template("temp.html",temp="*",hum="*")
```

Fonte: Elaborado pelo autor.

4.12 Algoritmo de Visualização da Webcam

O sistema para a visualização da *webcam* foi desenvolvido com a finalidade de tornar possível monitorar o local onde foi instalada. O acesso se dá através de um navegador de internet de qualquer dispositivo da rede local, ou até mesmo através da internet, caso a rede permita conexões remotas. O algoritmo foi desenvolvido com uma implementação do OpenCV, que utiliza a classe “VideoCapture” para receber os quadros do vídeo da *webcam*, retornando uma *stream* de vídeo para a página de controle de acesso do *website*. Esta conexão se dá através do micro *framework* Flask, que através de uma implementação de WSGI permite a interação da linguagem Python para o desenvolvimento de aplicações *web*.

Figura 43 – Algoritmo de visualização da webcam

```
class VideoCamera(object):
    def __init__(self):
        # Open a camera
        self.cap = cv2.VideoCapture(-1)
        if not self.cap.isOpened():
            raise RuntimeError('A camera nao pode ser iniciada!')
        # Initialize video recording environment
        self.is_record = False
        self.out = None

        # Thread for recording
        self.recordingThread = None

    def __del__(self):
        self.cap.release()
        print("A camera esta sendo desabilitada...")

    def get_frame(self):
        ret, frame = self.cap.read()

        if ret:
            ret, jpeg = cv2.imencode('.jpg', frame)
            return jpeg.tobytes()
        else:
            return None
def video_stream():
    global video_camera
    global global_frame

    if video_camera == None:
        video_camera = VideoCamera()

    while True:
        frame = video_camera.get_frame()

        if frame != None:
            global_frame = frame

            yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        else:
            yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n' + global_frame + b'\r\n')

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

Fonte: Elaborado pelo autor.

4.13 Algoritmo de Log de Temperatura e Umidade

O sistema para o envio de log de dados do sensor de temperatura e umidade AM-2302 para a base de dados foi desenvolvida para que a cada período (10 minutos), seja efetuada uma leitura do sensor e armazenagem dos dados. Essa programação sistemática é necessária para que se crie um modelo simétrico entre os dados e o tempo, para que se possa ter uma abordagem mais realística no que se diz respeito ao controle e manipulação dos dados.

Figura 44 – Algoritmo que armazena logs de temperatura e umidade

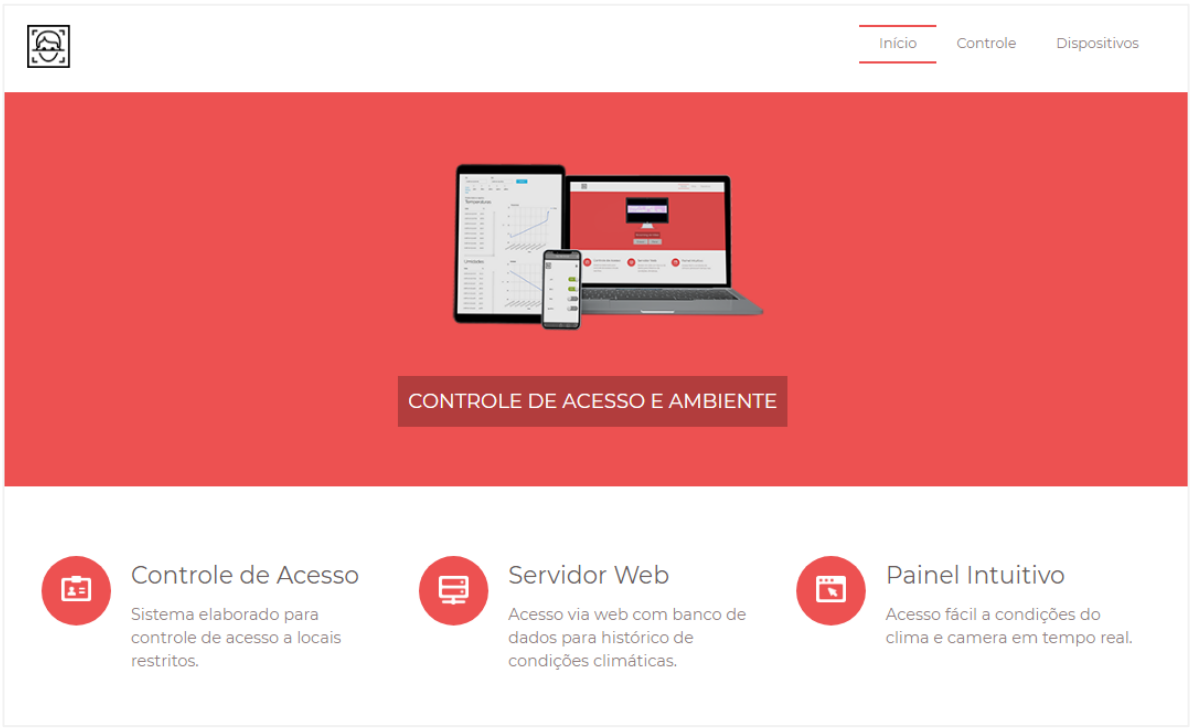
```python
import sqlite3
import sys
import time
import Adafruit_DHT

def log_values(sensor_id, sensor_id2, temp, hum):       # função que envia os dados para o database
    conn=sqlite3.connect('/var/www/tcc/tcc_app.db')     # arquivo do banco de dados

    curs=conn.cursor()
    curs.execute("""INSERT INTO temperatures values(datetime(CURRENT_TIMESTAMP, 'localtime'),
        (?), (?))""", (sensor_id,temp))
    curs.execute("""INSERT INTO humidities values(datetime(CURRENT_TIMESTAMP, 'localtime'),
        (?), (?))""", (sensor_id2,hum))
    conn.commit() # executa os comandos com o SQlite 3
    conn.close()

while True:                                                                 # loop
    humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4) # recebe os dados
    if humidity is not None and temperature is not None:                    # se não for vazio
        log_values("1", "2", temperature, humidity)                         # envia os dados
    else:                                                                   # se for vazio
        log_values("1", "2", 0, 0)                                          # envia nulo
    time.sleep(180)                                                         # 3 minutos de espera
```

Fonte: Elaborado pelo autor.

5 INTERFACE

A figura 45 apresenta a tela inicial do módulo web, na qual é apresentado um menu superior contendo três opções:

Figura 45 – Interface da página inicial do website



Fonte: Elaborado pelo autor.

- Item "**Início**" do menu: Redireciona para a página principal do website.
- Item "**Controle**" do menu: A página é redirecionada para o controle de temperatura umidade e acesso.
- Item "**Dispositivos**" do menu: A página é redirecionada para a página de controle de dispositivos.

5.1 Interface do Controle de Dispositivos

A figura 46 apresenta a interface da página de controle de dispositivos, onde é possível realizar o acionamento e desligamento dos relés, através dos botões deslizantes que aparecem na tela com a informação de "on" para ligado ou "off" para

desligado. Abaixo da tela é apresentada uma informação contendo um botão que quando pressionado redireciona diretamente para a página onde há a visualização de temperatura e umidade em tempo real.

Figura 46 – Interface da página de controle de dispositivos

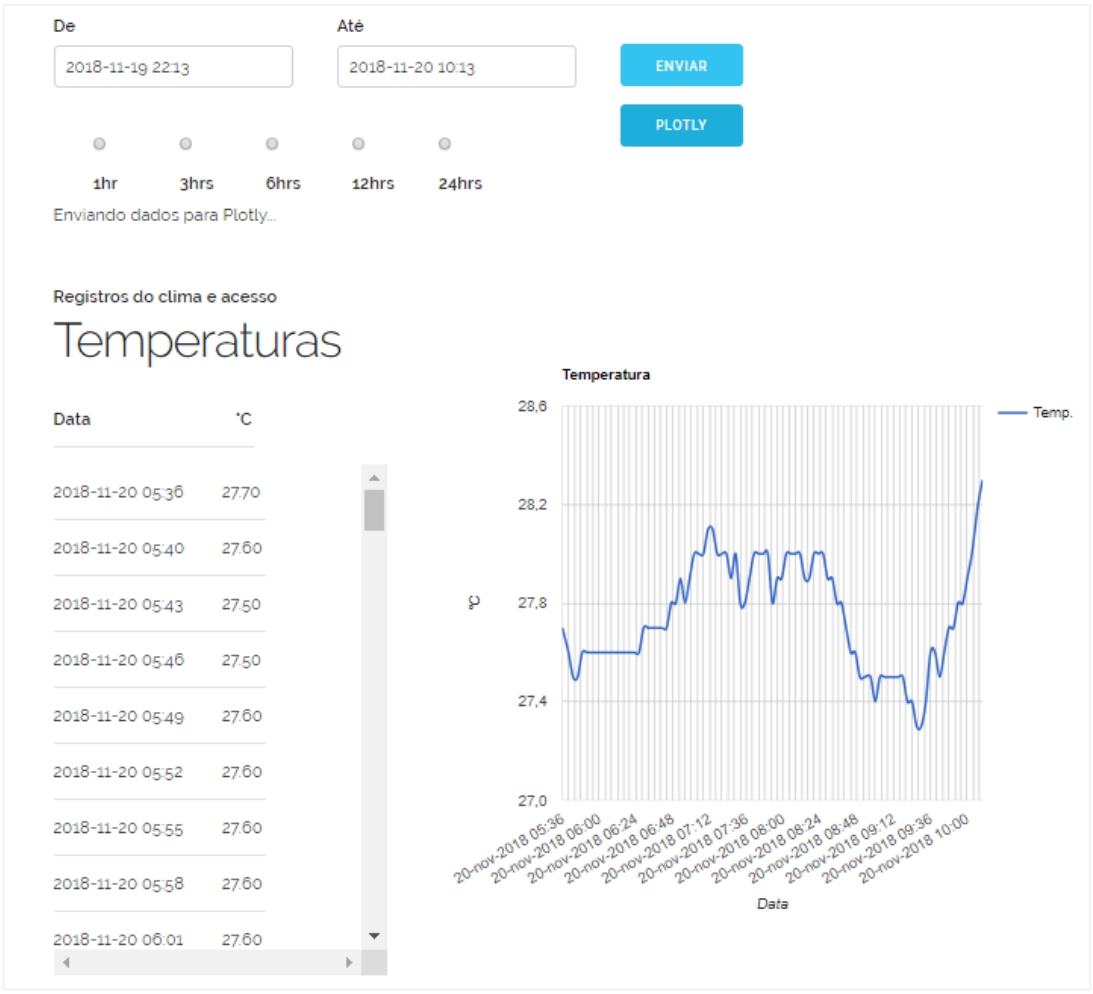
Fonte: Elaborado pelo autor.

- Botão deslizante na posição **"on"**: Liga o dispositivo conectado ao canal correspondente.
- Botão deslizante na posição **"off"**: Desliga o dispositivo conectado ao canal correspondente.
- Botão **"Acessar Dados"**: A página é redirecionada para a visualização de temperatura e umidade em tempo real.

5.2 Interface do Controle de Temperatura e Umidade e Acesso

A figura 47 apresenta a interface da página de controle de temperatura, umidade e acesso, esta página oferece a visualização e filtragem dos dados de forma intuitiva, que também conta com a visualização da câmera interna para o monitoramento do ambiente.

Figura 47 – Interface da página de controle de temperatura, umidade e acesso



Fonte: Elaborado pelo autor.

- Caixa de texto “**Data inicial**”: Ao clicar é aberto um componente datetimepicker desenvolvido em JQuery, que melhora a usabilidade para a escolha da data inicial e funciona também como forma de validação.
- Caixa de texto “**Data final**”: Ao clicar é aberto um componente datetimepicker, que melhora a usabilidade para a escolha da data final e funciona também como forma de validação.
- Botão “**Enviar**”: Ao clicar no botão é realizada a pesquisa de temperatura, umidade e acesso que estão armazenadas no banco de dados e retorna em uma lista.
- Listas de dados: Ao lado esquerdo serão exibidas listas de dados correspondentes a temperatura, umidade e acesso. Pode-se utilizar a barra de rolagem para ver todos os itens da lista caso necessário.

- Botão “**Plotly**”: Ao clicar os dados presentes na lista de pesquisa de temperatura e umidade, são enviados para a API Plotly onde irá gerar um link.
- Link do Plotly: Após clicar no botão “Plotly” o link Plotly será gerado contendo o link para acesso ao gráfico, basta clicar no *link* para ser redirecionado para a página do *website* do Plotly que contém o gráfico, como pode ser visto na Figura 48:

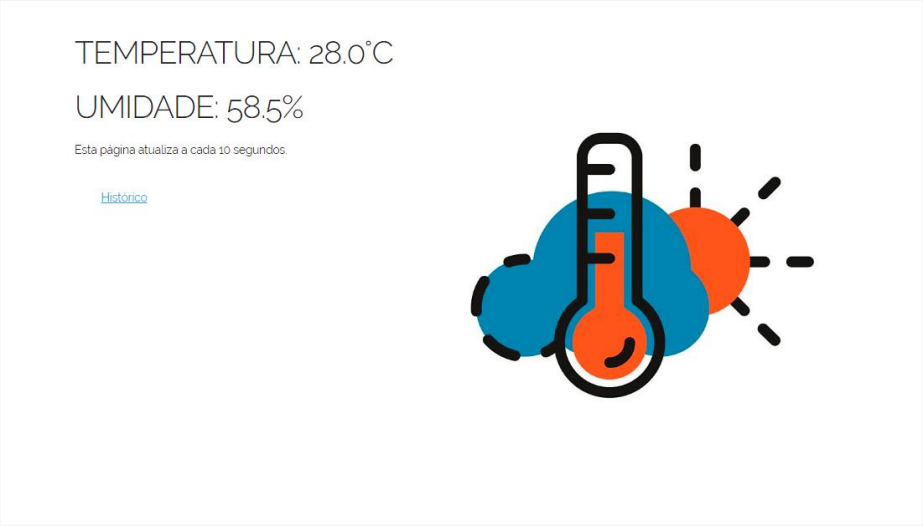
Figura 48 – Gráfico cruzando dados do sistema enviados ao Plotly

Fonte: Elaborado pelo autor.

5.3 Interface de Visualização de Temperatura e Umidade

A figura 49 apresenta a *interface* da página de visualização de temperatura e umidade, nesta tela é possível ver a temperatura e umidade do ambiente e ela é atualizada automaticamente a cada período de 10 segundos.

Figura 49 – Interface da página de controle de dispositivos



Fonte: Elaborado pelo autor.

- Link **“histórico”**: Redireciona para a página de controle de temperatura e umidade.

6 CONSIDERAÇÕES FINAIS

Este capítulo abordará como foi o processo de desenvolvimento do sistema e se o resultado atendeu ou não as expectativas iniciais. Para isso serão analisadas as funcionalidades do sistema desenvolvido neste projeto, a fim de comprovar suas funcionalidades de forma clara e objetiva.

O algoritmo para criação do conjunto de dados de faces funcionou da forma esperada, retornando diversas fotografias que somente o rosto é fotografado e convertido em escala de cinza, de forma que o conjunto de dados de imagens esteja pronto para a próxima etapa do processo de reconhecimento facial, que treinará as imagens gerando os LBPH de cada usuário. O algoritmo de treinamento se mostrou rápido e eficaz no treinamento das imagens, cumprindo com a função de criar histogramas das imagens da base de dados e associá-los ao ID do usuário.

Após a conclusão dos procedimentos de obtenção das imagens de faces em escala de cinza pelo algoritmo responsável por criar o conjunto de dados e o treinamento das imagens ser concluído gerando o LBPH de cada usuário, o algoritmo de reconhecimento facial foi capaz de reconhecer as faces dos usuários que tiveram suas imagens faciais treinadas pelo algoritmo e acionar a fechadura eletrônica através do acionamento dos relés, com a ajuda da assistência por voz implementada neste projeto, completando com êxito as funcionalidades que foram propostas.

Neste trabalho inicialmente foi necessário muito estudo das tecnologias agregadas a este projeto e a assuntos relacionados a área de automação de dispositivos e visão computacional.

Na sequência, apresentou-se todo o processo de desenvolvimento da parte de *hardware* e eletrônica do projeto, que incluiu a construção de uma caixa metálica para a montagem de todos componentes de forma limpa, segura e que bem acomodasse a instalação de sensores e componentes.

No desenvolvimento do sistema, houve algumas dificuldades dado a complexidade do desenvolvimento dos módulos e a ampla abordagem, com a necessidade de conhecimentos específicos das áreas. Uma das dificuldades que surgiu, foi a implementação conjunta de todas as funções no mesmo sistema, pois inicialmente cada função do sistema foi desenvolvida a parte tanto no módulo *web* quanto no *desktop*. Houve testes em cada uma das funcionalidades para garantir

que tivessem um correto funcionamento. A partir do momento que todas as funções estavam desenvolvidas, se deu início a implementação de todas elas de forma conjunta causando muitas incompatibilidades e imprevistos, porém, com muita pesquisa e dedicação foi possível alinhar e integrar todas as funcionalidades dentro do que foi proposto e obter sistemas funcionais.

A interface do usuário foi construída a fim de manter uma boa experiência do usuário, seguindo os padrões utilizados e valorizados no mercado, como cores flat e a responsividade do *website,* adaptando-se a dispositivos com diferentes tamanhos de tela.

O sistema cumpre o que promete, pois supriu todos os requisitos que haviam sido propostos incialmente, entre eles, o reconhecimento facial para o controle de acesso a ambientes restritos, controle de ambientes onde é possível obter dados de temperatura, umidade e acesso, visualização da câmera interna, comunicação com a API do Plotly, assistência por voz através da biblioteca Text To Speech e controle de dispositivos remotamente.

Este trabalho teve como objetivo final, o desenvolvimento de sistemas funcionais de controle de acesso por reconhecimento facial e controle de ambientes, e isso foi alcançado, como visto neste capítulo, de forma até mais completa do que havia sido pensado inicialmente.

Como possíveis trabalhos futuros, pode-se apontar o desenvolvimento de visão computacional aplicado a outras áreas, como na identificação de objetos, contagem de pessoas, entre outros. Como este projeto ainda conta com uma boa capacidade de expansão, poderão ser feitas instalações de novos sensores e dispositivos, como por exemplo, novos relés que possuam maior quantidade de canais ampliando a quantidade de dispositivos que poderão ser ligados e desligados pelo sistema e também aperfeiçoar ainda mais o controle de temperatura e umidade com a instalação de sensor de infravermelho para controlar aparelhos de ar condicionado.

REFERÊNCIAS

Ahonen, T., Hadid, A., and Pietikainen, M. **Face Recognition with Local Binary Patterns. Computer Vision**. ECCV 2004 (2004), p. 469–481.

Alexander Kuranov, Rainer Lienhart, and Vadim Pisarevsky. **An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of Haar-like Features**". Intel Technical Report MRL-TR-. Julho, 2002.

AMAZON: **Amazon Rekognition**. 2018. Disponível em: <https://aws.amazon.com/pt/rekognition/>. Acesso em: 08 nov. 2018.

Aosong Electronics. Datasheet: **AM2303 Humidity and Temperature Module.** Electronic Publication, 2015.

B. Froba and A. Ernst, "**Face detection with the modified census transform**," in IEEE Int. Conf. on Automatic Face and Gesture Recognition, 2004, p. 91–96.

Caifeng Shan, Shaogang Gong, and Peter W. McOwan, "**Facial expression recognition based on Local Binary Patterns: A comprehensive study**," Image and Vision Computing, vol. 27, no. 6, p. 803–816, 2009.

Freund, Yoav; Schapire, Robert E. **A decision-theoretic generalization of on-line learning and an application to boosting**. Journal of Computer and System Sciences, 1997, p. 119-139.

GOOGLE: **Google Fotos. 2018**. Disponível em: <https://www.google.com/photos/about/>. Acesso em: 18 nov. 2018.

JSON: **Introducing JSON.** 2018. Disponível em: <https://www.json.org/>. Acesso em: 20 out. 2018.

MEDIUM: **Object Detection Using Local Binary Patterns**. 2017. Disponível em: <https://medium.com/@ckyrkou/object-detection-using-local-binary-patterns-50b165658368>. Acesso em: 20 out. 2018.

NAMAN: **Experimenting with Kinect using opencv, python and open kinect**. 2014. Disponível em: <https://naman5.wordpress.com/2014/06/24/experimenting-with-kinect-using-opencv-python-and-open-kinect-libfreenect/>. Acesso em: 17 out. 2018.

OPENCV - Open Source Computer Vision: **Cascade Classification**. 2018. Disponível em: <https://docs.opencv.org/3.0-beta/modules/objdetect/doc/cascade_classification.html>. Acesso em: 15 de novembro. 2018.

OPENCV - Open Source Computer Vision: **Face Detection using Haar Cascades**. 2017. Disponível em:

<https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html>. Acesso em: 15 out. 2018.

OPENCV - Open Source Computer Vision: **Face Recognition with OpenCV**. 2018. Disponível em: <https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html>. Acesso em: 17 out. 2018.

Paul Viola and Michael J. Jones, "**Rapid Object Detection using a Boosted Cascade of Simple Features**", IEEE CVPR, 2001.

PYPI: **RPi.GPIO 0.6.5**. 2018. Disponível em: <https://pypi.org/project/RPi.GPIO/>. Acesso em: 18 out. 2018.

PYTHON: **PEP 333 - Python Web Server Gateway Interface v1.0. 2003**. Disponível em: <https://www.python.org/dev/peps/pep-0333/>. Acesso em: 08 out. 2018.

Rainer Lienhart, Alexander Kuranov, Vadim Pisarevsky. **Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection**. Intel Technical Report MRL-TR. maio, 2002.

RASPBERRY Pi: **RASPBERRY PI 3 MODEL B+**. 2018. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>. Acesso em: 07 nov. 2018.

ROSEBROCK, Adrian. Pyimagesearch: **Local Binary Patterns with Python & OpenCV**. 2015. Disponível em: <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>. Acesso em: 08 dez. 2018.

SQLITE: **About SQLite**. 2018. Disponível em: <https://www.sqlite.org/about.html>. Acesso em: 04 nov. 2018.

SOMMERVILLE, I. **Engenharia de Software**. 8ª Edição. Editora: Pearson Addison-Wesley. São Paulo, 2007.

TOWARDS Data Science: **Face Recognition: Understanding LBPH Algorithm**. 2017. Disponível em: <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>. Acesso em: 19 out. 2018.

VISHAY. Datasheet: **74HC4051 16 x 2 Character LCD**. Electronic Publication, 2002.

WIKIPEDIA: **Kinect**. 2018. Disponível em: <https://en.wikipedia.org/wiki/Kinect>. Acesso em: 10 nov. 2018.

WIKIPEDIA: **Application programming interface**. 2018. Disponível em: <https://en.wikipedia.org/wiki/Application_programming_interface>. Acesso em: 20 nov. 2018.