Câmera de segurança com detecção de presença usando Raspberry Pi

Gustavo Luiz Monteiro de Oliveira
Estudante de Engenharia Eletrônica
pela Universidade de Brasília
Matrícula: 150128673
Email: gustavo.luiz@aluno.unb.br

Lucas Pereira Pires
Estudante de Engenharia Eletrônica
pela Universidade de Brasília
Matrícula: 170108996
Email: pereira.pires@aluno.unb.br

***Abstract*—O foco deste relatório é o desenvolvimento de um sistema de câmera de segurança, com detecção de presença corporal e facial. O projeto utilizará uma *Raspberry Pi* e um módulo de câmera, com o algoritmo na linguagem Python, utilizando a biblioteca OpenCV[1] para a implementação da detecção de presença. O sistema deverá ser capaz de fornecer a imagem em tempo real através de um servidor local, além de notificar remotamente o usuário em caso de presença no campo de visão da câmera.**

I. Introdução

O uso do *IoT* para consumidores é frequentemente chamado de mercado doméstico inteligente, e grande parte desse mercado consiste em dispositivos de segurança doméstica. Os consumidores geralmente são motivados a comprar dispositivos de segurança doméstica inteligentes para evitar roubos [2]. Neste sentido, a informação é um ponto essencial para a tomada de decisões em situações criticas, seja para o acionamento de autoridades em situações de possível invasão, como para uma checagem prévia da identidade de uma possível visita.

Somente o mercado global de dispositivos de segurança doméstica inteligente foi de US$ 2,14 bilhões em 2018, e espera-se que aumente para cerca de US$ 5,05 bilhões até 2025 [3]. O dispositivo de segurança doméstica mais comum é a câmera [4].

O objetivo deste projeto é fornecer um sistema de vigilância simples e de baixo custo, em contraste com sistemas disponíveis no mercado, visando um público-alvo específico, que são os proprietários de residências, preferencialmente casas, que desejam obter visualização em tempo real e com a utilização de visão computacional para notificá-los de uma possível presença detectada no campo de visão da câmera.

II. Desenvolvimento

A. Descrição do Hardware

O *hardware* do projeto consistirá de basicamente um Raspberry Pi acompanhado de um módulo de câmera para Raspberry Pi. Além disto será utilizado um cabo Ethernet para conexão com a rede local para fornecimento das imagens através do servidor e se disponível conexão externa, realizar a notificação do usuário quando uma presença for detectada.

1) Placa Raspberry Pi: um mini-computador, criado com o foco na educação de programação de computadores, mas que é utilizada em diversos tipos de projetos por sua versatilidade e historicamente um baixo custo. Este computador roda um sistema Linux diretamente de um cartão SD e possui múltiplas formas de conexão externa, desde pinos GPIO até porta ethernet e portas USB e HDMI.

Foi citado anteriormente que historicamente o Raspberry Pi é conhecido pelo seu baixo custo, recentemente foi anunciado [5] pelo CEO da Raspberry Pi Ltd., Eben Upton, o primeiro aumento no preço das placas devido à escassez de chips provocada pela pandemia da COVID-19 [6]. Na prática não se encontram placas disponíveis nas distribuidoras e os preços oferecidos por revendedores são até 4 vezes maiores que o preço sugerido pela Raspberry Pi Foudation. Apesar disso, o estudo de viabilidade da aplicação do Raspberry Pi no projeto foi feito considerando uma situação fora da crise de fornecimento.

O modelo utilizado será o Raspberry Pi 4B 2GB, com as principais especificações para o desenvolvimento do projeto listadas a seguir:

- Processador Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz;
- RAM 2GB LPDDR4-3200 SDRAM;
- WiFi 2.4 GHz e 5.0 GHz IEEE 802.11ac;
- Porta micro-HDMI;
- Porta MIPI CSI para câmera;
- *Slot* para cartão Micro-SD;
- Alimentação 5V DC via USB-C.

2) Módulo de Câmera para Raspberry Pi:

- Sensor de 5MP, modelo OV5647;
- Vídeo em 1080p a 30 fps com codec H.264 (AVC);
- Conexão CSI.

B. Descrição do Software

1) Código preliminar em Python*:* Para o código inicial, foi utilizado o código Smart Security Camera [10] como referência. O arquivo *main.py* usa o *flask*, que é um *framework*

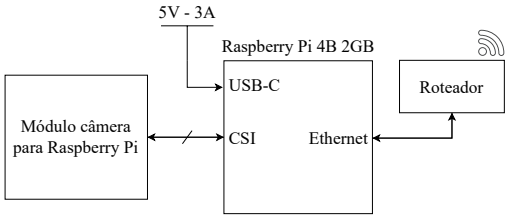

Fig. 1. Esquemático do Sistema.

TABLE I. BOM - BILL OF MATERIALS.

Nome	Descrição	Preço
Raspberry Pi 4B 2GB	Componente Eletrônico	R$ 639,90 [8]
Módulo de Câmera para Raspberry Pi	Componente Eletrônico	R$ 78,90 [9]
Plástico ABS para impressão 3D	Matéria-prima para o suporte	A definir
	TOTAL	R$ 718,80

web Python mais leve que fornece recursos que facilitam a criação de aplicativos *web* em *Python* (Ver Fig.4).

Primeiramente é executada a interface com o módulo de câmera da *Raspberry*, que permite fornecer a imagem em tempo real através da biblioteca *PiVideoStream*, e são utilizadas as funções da biblioteca *OpenCV* para fazer a detecção da pessoa através de um dos modelos provenientes de um banco de dados da biblioteca.

Em seguida, a imagem é lida *frame* a *frame* e o classificador de imagem é utilizado com o objetivo de conseguir identificar a pessoa na imagem e, para isso, alguns padrões como tamanho, cor e fator de escala são estabelecidos. Uma vez que a pessoa é detectada, um retângulo é desenhado ao redor da mesma e a imagem modificada é retornada.

Nos testes iniciais, o processo de detecção funcionou apenas utilizando o modelo de detecção facial, como visto na Fig.2.

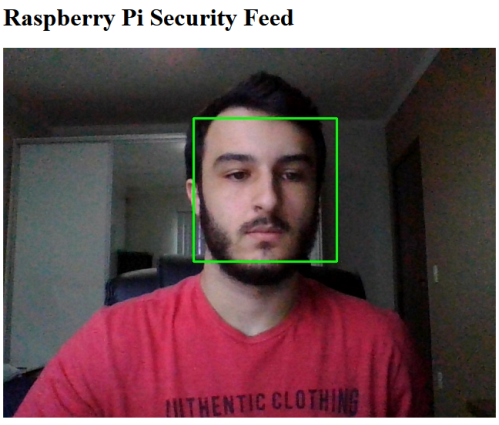

Fig. 2. Resultado da detecção facial com *webcam*.

Em seguida, a imagem é comprimida na extensão *jpg* e codificada em um *buffer* de memória. Em seguida esse *buffer* é acessado e os dados da imagem como uma *string* contendo os *bytes* de cada *frame* são retornados.

A segunda parte é relativa ao envio em si de uma mensagem via *e-mail* para o usuário com a imagem da pessoa detectada e o *link* para acompanhar em tempo real a imagem através de um servidor local.

Foi criado, portanto, o arquivo *mail.py*, como visto na Fig. 4. O código utiliza o módulo *smtp* da biblioteca *smtplib* para que seja possível criar uma conexão com o servidor do *Gmail* que realiza o *login* na conta que vai enviar a mensagem para o destinatário quando uma pessoa for detectada pela camêra da *Raspberry*.

O *e-mail* consiste em um cabeçalho com os textos ”De”, ”Para”, ”Assunto” e o corpo da mensagem, e por esses cabeçalhos serem, por padrão, textos ASCII simples, é utilizado o *MIME*, uma extensão que permite especificar o tipo de conteúdo no *e-mail*, por exemplo, se é um texto ou uma imagem, ou ambos, e os agrupa no corpo do *e-mail*.

Fig. 3. *E-mail* enviado ao destinatário contendo a imagem da pessoa detectada.

Como a mensagem neste projeto tem mais de um tipo de conteúdo, foi utilizado o formato *MIMEMultipart* pra representar uma estrutura que permite anexar esses conteúdos no corpo do *e-mail* através de subclasses que criam objetos do tipo *MIMEText*, que armazena o texto da mensagem, e o *MIMEImage* que armazena e anexa a imagem em si, ambos inseridos em um template *HTML*, como pode ser verificado na Fig.3.

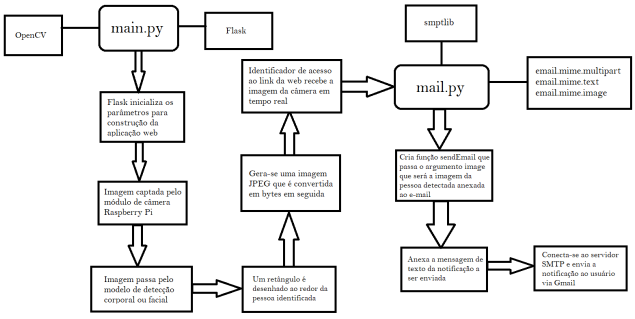

Fig. 4. Fluxograma dos códigos preliminares em *Python*.

C. Código Definitivo

Para o código final do projeto, é proposto uma divisão entre dois principais processos. A parte de captura e processamento de imagem, detecção de objetos e temporização está sendo

desenvolvida em C++, utilizando a biblioteca de visão computacional, *OpenCV* e as imagens são capturadas a partir do módulo de câmera da *Raspberry* e não mais da *webcam*, como feito anteriormente em *Python*. Para fornecimento de vídeo em tempo real da câmera, um segundo processo é iniciado, a partir de um emphscript em *Javascript*, executado em *Node.js*, que fornece uma página HTML na rede local com as imagens atualizadas.

O fornecimento da imagem é feito de forma que o código em C++ captura, processa e salva a imagem em um arquivo na pasta local e o servidor (*Node.js*) mostra esta imagem em uma página da *web* através de um *socket* conectado ao navegador (cliente). Os testes iniciais com o código em C++ revelaram a necessidade de alguns ajustes no processo de comunicação entre servidor e o programa principal, para uma experiência satisfatória na visualização. Para isto, foi então adicionada uma camada IPC (*Inter-Process Communication*) entre os dois processos, utilizando um *socket* local para que o programa principal, escrito em C++, mande um sinal no momento em que o servidor deve ler e propagar a imagem capturada.

Para melhor desempenho do sistema desenvolvido, o código em C++ foi dividido em duas *threads* secundárias, além de um temporizador utilizando o sinal *SIGALRM* para ajuste da taxa de captura e transmissão. A primeira *thread* fica responsável pela captura e processamento da imagem, salvando o resultado em memória e enviando um sinal para a próxima *thread*. A segunda *thread* mantém a conexão com o *socket* local, e quando recebe o sinal da primeira *thread*, salva a imagem que está armazenada na memória em um arquivo *.jpeg* e propaga o sinal ao servidor através do *socket* local.

O servidor executado em *Node.js* fornece a página HTML na porta 3000 e mantém as conexões com o *socket* local e os *socket*s dos clientes para, ao receber um sinal de imagem disponível, realizar a leitura deste arquivo e enviar um sinal aos clientes conectados com a imagem em *buffer* no formato de *string* codificada em base 64. Um *script* na página HTML mantém a conexão com o *socket* do servidor e quando recebe o *buffer* com a imagem codificada, atualiza a página com a imagem mais recente.

Tendo em vista que o objetivo do trabalho é conseguir realizar uma detecção eficaz do corpo inteiro da pessoa, uma vez que em situações de roubo da residência, seria muito improvável que o assaltante mostrasse sua face, a detecção facial realizada anteriormente não seria razoável. Como o principal benefício do projeto é detectar estas situações e notificar o proprietário da residência, os próximos passos incluem testar o sistema com novos modelos de detecção de corpo inteiro. Além disso, será implementado o envio de e-mail com a imagem capturada em anexo, por meio do endereço SMTP do servidor e a porta correspondente, com a estrutura de título e corpo da mensagem feita em HTML.

Link do repositório: SOE Projeto Final

REFERENCES

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] R. Chen *et al.*, "Intent to purchase IoT home security devices: Fear vs privacy," *Plos One Journal*, 2021.

[3] Statista. Smart home security market revenue worldwide from 2018 to 2025. [Online]. Available: https://www.statista.com/statistics/1056057/worldwide-smart-home-security-market-value/

[4] ADT. Home is where the smart is. [Online]. Available: https://www.adt.com/home-is-where-the-smart-is/

[5] E. Upton. Supply chain, shortages, and our first-ever price increase. [Online]. Available: https://www.raspberrypi.com/news/supply-chain-shortages-and-our-first-ever-price-increase/

[6] TechRepublic. Global chip shortage: Everything you need to know. [Online]. Available: https://www.techrepublic.com/article/global-chip-shortage-cheat-sheet/

[7] Raspberry Pi. Raspberry pi 4 tech specs. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[8] RoboCore. Raspberry Pi 4 2GB - model B Anatel. [Online]. Available: rasp:https://www.robocore.net/placa-raspberry-pi/raspberry-pi-4-2gb

[9] Mercado Livre. Módulo câmera para raspberry pi 5MP. [Online]. Available: https://produto.mercadolivre.com.br/MLB-1676002030-modulo-cmera-p-raspberry-pi-5mp-c-cabo-flat-cnota-fiscal-_JM

[10] A. Tainter. Smart-security-camera. [Online]. Available: https://github.com/HackerShackOfficial/Smart-Security-Camera/