

Estação de Segurança residencial

João Marcos Kubitschek Nascimento Oliveira
Engenharia Eletrônica
Universidade de Brasília
Brasília, Brasil
190109637@aluno.unb.br

Vinícius Mendes Martins
Engenharia de Software
Universidade de Brasília
Brasília, Brasil
211063265@aluno.unb.br

I. INTRODUÇÃO

A preocupação com a segurança residencial tem se tornado uma questão central para muitas pessoas, alimentada pela incerteza sobre o que acontece em suas casas e seus filhos quando estão ausentes e pela sensação de impotência em caso de incidentes. Em resposta a esse desafio, a indústria de equipamentos e sistemas de segurança tem experimentado um crescimento significativo, impulsionado pela introdução de tecnologias avançadas, como inteligência artificial e armazenamento em nuvem. No entanto, a falta de customização de proteção (alarmar quando algo considerado perigo ocorrer), juntamente com a ausência de conectividade eficiente com dispositivos externos, como smartphones, resulta em uma experiência de controle residencial fraca para o usuário. Isso, por sua vez, pode gerar um sentimento de frustração e inadequação.

No entanto, existem soluções para superar essa lacuna de falta de customização de proteção (alarmar quando algo considerado perigoso ocorrer), juntamente com a ausência de conectividade eficiente com dispositivos externos por meio de computadores de placa única (SBC), proporcionando uma interface unificada de vigilância e customização para o usuário. Ao adotar uma interface simples e integralizada, os usuários podem desfrutar de maior controle sobre a segurança de suas casas, acessando e monitorando facilmente todos os aspectos do sistema por meio de uma única plataforma intuitiva. Essa integração não apenas aumenta a eficiência e a conveniência, mas também promove uma maior tranquilidade para os moradores, sabendo que estão protegidos por um sistema coeso e responsável.

Alguns produtos já fornecidos pelo mercado são Axis Camera Station Secure Entry, que contém um conjunto de equipamentos de segurança como câmera, portas com tranca eletrônica, leitores de cartão e tudo isso unificado em uma única interface que está distribuída em múltiplos servidores. Outro produto é o Sistema QVR que contém um sistema de gravação e análise com gestos perigosos ou suspeitos que tem suporte a inteligência artificial, além de serviço de armazenamento em nuvem. E por fim, mais um produto é HP7, que um interfone com vídeo inteligente, que contém integração com portas/portões e detecção de movimento.

O objetivo deste projeto é planejar e implementar uma estação de segurança residencial para o cuidado de crianças, que incorpore as últimas tendências em tecnologia, particularmente a inteligência artificial. Uma parte essencial desse sistema será uma rede neural treinada para o detecção de pessoas e objetos considerados perigosos. Além disso, a estação de segurança oferecerá uma interface gráfica integrada (GUI) que se conecta diretamente à câmera de vigilância e ao smartphone registrado pelo usuário.

II. OBJETIVOS

- Reconhecimento de objetos
- Contagem de pessoas
- Monitoramento da casa
- Fornecer informações em tempo real
- Notificação de incidentes
- Interface integrada e unificada
- Garantir segurança

III. REQUISITOS

A. Requisitos Funcionais

- Disponibilizar uma interface gráfica
- Cadastro de objetos perigosos
- Tirar foto do local com objeto perigoso
- Notificar objeto perigoso pelo Telegram
- Realizar contagem de pessoas no local
- Notificar pelo Telegram se o número de pessoas mudar
- Exibir o vídeo da câmera na interface gráfica
- Conceder acesso do vídeo das câmeras ao vivo por streaming

B. Requisitos Não-Funcionais

- Armazenar fotos cadastradas no sistema de diretórios da estação
- Aceitar mais de um usuário do Telegram
- Oferecer notificação em tempo real
- Notificar o usuário através do aplicativo Telegram
- A notificação deve conter informações de dia e horário

IV. BENEFÍCIOS

Este projeto de "Babá eletrônica" apresenta uma série de benefícios notáveis para os usuários. Primeiramente, oferece uma camada de sistema integrado através de um computador

de placa única (SBC), que dar suporte a conectividades entre os mais diversos dispositivos de monitoramento residencial com o adicional de segurança por meio da aplicação de tecnologias avançadas, como inteligência artificial, aumentando a eficácia na detecção de incidentes.

Além disso, a interface gráfica intuitiva simplifica a configuração e operação do sistema, tornando-o acessível mesmo para usuários menos experientes em tecnologia. Esta facilidade de uso é complementada pela integração fluida com dispositivos de segurança residencial existentes, garantindo uma transição suave para os usuários que desejam atualizar seus sistemas.

Outro ponto importante é a capacidade de receber alertas em tempo real sobre eventos de segurança, proporcionando aos usuários a oportunidade de responder rapidamente a situações de emergência ou atividades suspeitas.

E por fim, também é possível realizar um monitoramento a distância da casa e assegurar os cuidados a criança em tempo real, uma vez que o usuário sinta a necessidade de saber como está a situação em sua residência.

V. REVISÃO BIBLIOGRÁFICA

A. Axis Camera Station Secure Entry

Produto de controle de acesso e gerenciamento por software da empresa Axis Communication que contém um conjunto de equipamentos de segurança, além de um interface com todos estes integrados. Link do produto Axis Camera Station Secure Entry.



Fig. 1. Câmera da Axis

B. Sistema QVR

É um sistema de vigilância da empresa QNAP que integra gravação e análise de gestos, que tem um suporte muito forte em inteligência artificial para pesquisa de movimentos e detecção de incidentes. Link do produto Sistema QVR.



Fig. 2. Sistema de segurança da QVR

C. HP7

Um interfone inteligente da empresa EZVIZ com suporte a vídeos, destravamento de porta/portões, wifi e detecção de movimento. Link do produto HPA7.



Fig. 3. Interfone HP7

D. Alarme Verisure

Sistema de alarme com registro de imagem da empresa Verisure que oferece monitoramento de alarme pelo celular, botão de pedido de ajuda a agentes de segurança e/ou emergência. Link do produto Alarme Verisure.



Fig. 4. Câmeras de segurança da Verisure

VI. METODOLOGIA

Após a definição do escopo do nosso projeto, ou seja, do que o sistema deverá realizar e fornecer de serviços aos usuários, bem como as suas restrições e qualidade, entramos na fase de definição da arquitetura do projeto, a sua divisão em subsistemas e os seus respectivos testes. Para que tenhamos uma orientação para o desenvolvimento do projeto. Na definição da arquitetura do projeto, definimos o hardware alvo do sistema,

sistema operacional onde o sistema irá ser desenvolvido, o paradigma de programação que será utilizado, a linguagem de programação e os framework's. E o resultado foi:

- Raspberry Pi 3, Model B;
- Raspberry Pi OS;
- Programação Orientada a Objetos (POO);
- C++ e Python (Python será utilizado para os testes dos subsistemas);
- Gtkmm, OpenCV, Crow,;

A arquitetura do projeto foi definida desta forma por alguns motivos, que são:

- 1) A escolha do uso da Raspberry Pi 3, Model 3 é pelo motivo do alto processamento computacional que as ferramentas de visão computacional necessitam, além de ser preciso um hardware capaz e pequeno.
- 2) Será utilizado o Raspberry Pi OS pela fácil e larga documentação desenvolvida pela comunidade, e pela facilidade de configuração e uso da Raspberry Pi 3 por ela.
- 3) O paradigma POO contém diversos benefícios, sendo ele a alta organização e o reaproveitamento de código.
- 4) O C++ por ser uma linguagem compilada, possui uma alta performance e baixo consumo computacional comparada com outras linguagens, além da grande quantidade de recursos que ela contém. Já o Python foi utilizada na etapa de testes pela sua facilidade de desenvolvimento rápido e de fácil entendimento, além também da grande quantidade de documentação que possui.
- 5) O Gtkmm é um Framework de interface gráfica feito para C++ com compatibilidade com o sistema operacional Linux e com um bom gerenciamento de memória; O OpenCV é um Framework de visão computacional que diversos recursos, sendo os principais para o nosso projeto a aquisição de imagens pela webcam e a utilização de rede neural; E o Crow é um Microframework para web, que permite a criação de serviços HTTP e Webscoket, que será utilizado para a transmissão;

Como foi dito anteriormente, foi realizada a divisão do projeto em subsistemas para a realização dos testes básicos para verificar a viabilidade do objetivo do sistema. Os subsistemas divididos são:

- Acesso/Configuração via interface gráfica;
- Aquisição de imagem ;
- Envio de mensagens via Telegram;
- Disponibilizar transmissão ao vivo das imagens;
- Identificação de objetos;
- Identificação de rostos;

Unindo todos esse subsistemas, conseguimos visualizar a ideia do projeto, que é uma Babá eletrônica que tem como principal função, a identificação de itens perigosos e de contagem de pessoas em uma casa, e reportar aos usuários estes eventos. Esses subsistemas serão detalhados posteriormente, junto com a explicação de seus funcionamentos.

VII. DESCRIÇÃO DE HARDWARE

A. BOM

Um dos objetivos do projeto é conseguir desenvolver um protótipo com um custo inferior aos modelos vendidos comercialmente e de maneira livre, onde seja possível ter total acesso a segurança da criança sem precisar comprar diversos produtos, ou seja, ele se complementa apenas por periféricos de mais fácil acesso.

O custo para fazer o protótipo é apresentado na Tabela I, abaixo:

TABLE I
CUSTO DO DESENVOLVIMENTO DO PROTÓTIPO

Item	Quantidade	Preço
Raspberry PI3	1	R\$400,00
Monitor	1	R\$150,00
Mouse	1	R\$29,80
Teclado	1	R\$38,66
Câmera	1	R\$44,99
Fonte 5V/3A com cabo Micro USB	1	R\$29,90
		TOTAL
		R\$693,35

B. Esquemático

Para dar seguimento ao projeto, foi desenvolvido um esquemático como mostrado na figura 5 para mostrar as conexões entre os componentes utilizados.

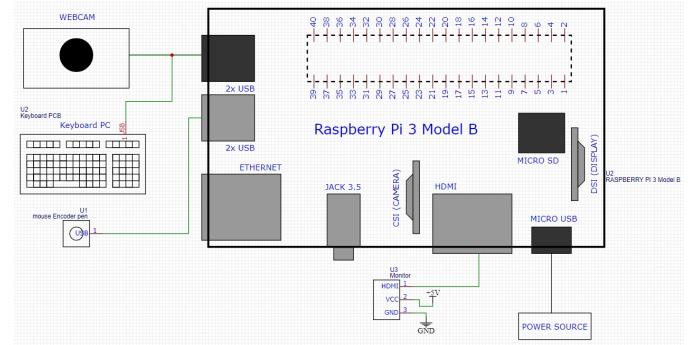


Fig. 5. Esquemático do projeto

O protótipo desenvolvido para o projeto da babá eletrônica pode ser resumido no uso da Raspberry PI 3 modelo B que servirá como nosso processador central conectada a periféricos. Sendo eles: Um teclado, um mouse e um monitor para o uso da interface gráfica; Uma câmera para receber informações que serão analisadas pelos programas que serão rodados na Raspberry. E, por fim, uma fonte para alimentar a Raspberry. O teclado, o mouse e a câmera são periféricos conectados pelas portas USB, o monitor será conectado na porta HDMI e a fonte de alimentação será conectada por uma porta micro USB.

C. Diagrama de blocos

O seguinte diagrama na figura 6, apresenta uma visão geral da organização do projeto.

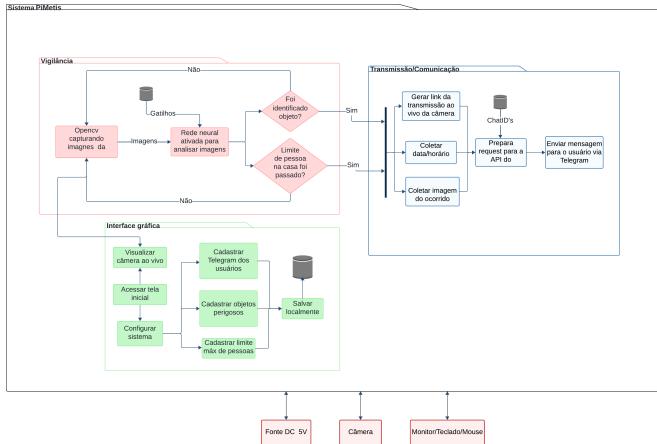


Fig. 6. Diagrama de Blocos do Sistema

Em resumo, o sistema foi dividido em três partes, na qual uma será responsável pela aquisição de imagens e analisar essas imagens para a identificação de objetos perigosos e a contagem de pessoas na casa.

A segunda parte é responsável em se comunicar com o usuário via Telegram, para reportar algum evento considerado perigoso, sendo enviado nesse reporte algumas informações úteis como data/horário do ocorrido, transmissão ao vivo da câmera e uma foto do ocorrido.

Por fim, na terceira parte, é definido uma interface gráfica para a configuração básica do sistema e para a visualização ao vivo da câmera.

VIII. DESCRIÇÃO DE SOFTWARE

A. Interface Gráfica

Como foi dito na metodologia, para o desenvolvimento da interface gráfica, escolhemos a biblioteca Gkmm que é uma interface em C++ do GTK+, que possui uma biblioteca completa de callbacks, sinais e um conjunto de widgets. Neste teste foi feita uma integração entre o Gkmm e o OpenCV, na qual utilizamos alguns recursos do sistema que foram mutex e thread para que fosse executado a interface gráfica ao mesmo tempo da aquisição de imagens pelo OpenCV. Primeiro de tudo foi feito um widget do tipo imagem onde vai ser colocado as imagens da câmera, e para isso é utilizado o mutex para garantir que nenhuma thread esteja configurando este widget ao mesmo tempo. Após isso é criado uma thread para a aquisição de imagens da câmera que é uma função em loop, na qual é feito a leitura da câmera pelo OpenCV.

B. YOLO

O YOLO (You Only Look Once) considerado uma técnica estado da arte em detecção de objetos, é um modelo de reconhecimento de objetos em tempo real, que faz a detecção e classificação de classes (objetos) em uma única passagem

pela rede neural. O algoritmo do YOLO pode ser dividido em 7 passos, que em resumo é a passagem de uma imagem por uma rede neural convolucional, onde as características serão extraídas e passadas por camadas conectadas para a previsão de classes e coordenadas de caixas delimitadoras, a imagem é então dividida em uma grade células que representam caixas delimitadoras e a probabilidade de uma classe, a partir disso é utilizada um filtro de supressão de não-máximo para filtrar caixas delimitadoras sobrepostas, e o resultado final é uma lista de caixas delimitadoras previstas e um rótulo da classe para cada objeto na imagem. Este processo será apresentado na figura x.

No modelo YOLO v4, que é o utilizado no teste, possui técnicas que a torna uma das versões mais importante. Uma das técnicas é o "Spatial Pyramid Processing" (SPP) que permite a extração de características de imagens de diferentes tamanhos e escalas. A outra técnica é a "Cross-stage partial connection" (CSP) que melhora a precisão do modelo usando uma combinação de múltiplos modelos e diferentes arquiteturas e escalas, assim combinando as suas previsões para conseguir uma precisão melhor. A partir disto, nos utilizamos um modelo pequeno pré-treinado do YOLOv4 e o seu peso, que consegue detectar 80 classes, que são objetos que são vistos diariamente (ônibus, avião, pessoa, cadeira, faca, etc.). Com isso nós utilizamos a webcam para fornecer as imagens de entrada para o modelo, e quando houver a detecção de alguma classe, é desenhado uma caixa ao redor com o seu nome e a porcentagem de precisão, como pode ser visto na figura 7 a seguir.

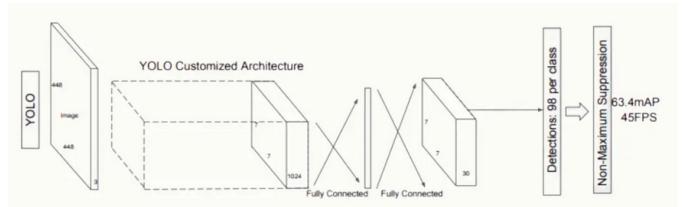


Fig. 7. Estrutura do modelo YOLO. Fonte: Wei Liu, Dragomir Anguelov, Dumitru Erha, Christian Szegedy, Scott Reed, Cheng-Yang Fu1, Alexander C. Berg. SSD: Single Shot MultiBox Detector.

C. Telegram Bot API

A API do Telegram Bot é uma ferramenta para criar bots no Telegram. Com ela, é possível criar bots que interagem com usuários, enviando mensagens, fotos e outras tipos de mensagens. No caso de teste que foi desenvolvido, foi necessário criar um novo bot com o seu token de identificação, e após isso foram feitos testes de envio de mensagens de textos e imagens para um usuário que interagiu com o bot.

D. Flask e ngrok

Flask é um Microframework leve em Python para desenvolvimento web. Utilizado no teste de transmissão para a criação de uma página web com um template com um simples HTML que espera uma imagem, que é preenchida com imagens capturadas pelo OpenCV.

Ngrok é uma ferramenta que cria túneis seguros para expor servidores locais à internet, permitindo compartilhar temporariamente aplicativos web hospedados localmente para testes ou demonstrações. Com o Flask criando um servidor local https, o Ngrok é utilizado para expor esta porta para a internet, através de um link.

E. Recursos do Sistema Operacional

Mutex, signal, thread e temporizador são recursos importantes dos sistemas operacionais e que podemos utilizar em nossos sistemas. O mutex é um mecanismo de sincronização que garante que apenas um thread ou processo acesse uma seção crítica de código ou recurso compartilhado por vez, evitando acesso mútuo. O sinal é uma interrupção usada para comunicação entre processos ou notificação sobre eventos específicos. A thread é a menor unidade de execução que pode ser agendada pelo sistema operacional, permitindo que um processo execute múltiplas tarefas simultaneamente dentro do mesmo espaço de endereçamento, podendo realizar tarefas de propósitos diferentes ao mesmo tempo. O temporizador é uma ferramenta usada para medir intervalos de tempo ou agendar a execução de código após um período específico, podendo ser de hardware ou software, e é utilizado para funções como pausar a execução de threads ou executar tarefas periodicamente. Esses recursos garantem a sincronização, comunicação e gerenciamento eficaz de processos e threads no sistema operacional.

F. Integração entre os subsistemas

O software integra diversos subsistemas utilizando a programação orientada a objetos (POO). Essa abordagem facilita a modularização do código, permitindo que cada subsistema seja desenvolvido e mantido como uma entidade independente, mas com interfaces bem definidas para interação. A POO promove reutilização de código e encapsulamento, garantindo que a comunicação entre os subsistemas seja eficiente e escalável, resultando em um sistema robusto e de fácil manutenção.

G. Cmake

A utilização do CMake para a compilação do sistema garante uma construção eficiente e configurável em múltiplas plataformas. CMake é uma ferramenta poderosa que automatiza a geração de scripts de compilação, facilitando a integração de bibliotecas externas e a gestão de dependências. Com CMake, foi possível simplificar o processo de construção, tornando-o mais consistente e adaptável às necessidades específicas do projeto. Isso resulta em um desenvolvimento mais ágil e uma maior portabilidade do código-fonte.

IX. RESULTADOS

A. Ponto de Controle I: Python

Com os subsistemas já divididos, e feita a descrição dos algoritmos utilizados, será apresentado o resultado dos testes de cada subsistema que foram feitos em Python para a verificação da viabilidade do desenvolvimento do projeto babá eletrônica.

1) *Teste da Interface Gráfica:* O teste realizado da interface gráfica que será utilizado para usuário para a configuração de simples como introduzir o sistema ao usuário e o apresentar o acesso ao bot do Telegram, como também na configuração de um parâmetro do sistema (a definição de quais objetos são considerados perigosos). Além disto, a principal tarefa que é oferecer a imagem da câmera ao vivo pela interface gráfico, e o teste foi focado em cima desta função para saber se é viável está funcionalidade, e depois, se fosse possível produzir esta funcionalidade na Raspberry Pi.

O teste foi desenvolvido inteiramente em linguagem C++, utilizando as bibliotecas já citadas na metodologia, que são o Gtkmm e o OpenCV, a figura 8 a seguir demonstra o resultado do teste.

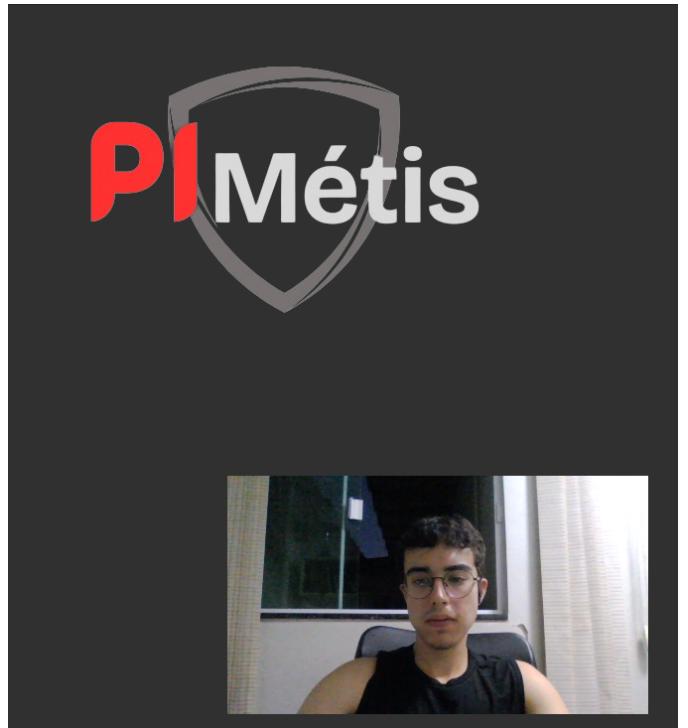


Fig. 8. Resultado do teste da interface gráfica. Fonte: Autores

2) *Teste do YOLO:* O teste realizado do YOLO tem como principal função verificar a viabilidade de utilizar este modelo pré-treinado para detecção de diversas classes, sendo destas a principal, uma pessoa, para que seja possível sanar um dos requisitos funcionais do sistema que é contabilizar a quantidade de pessoas.

O teste foi desenvolvido em Python 3, devido a curva de aprendizagem e a praticidade, sendo está o primeiro contato da equipe com o YOLO. A figura 9 a seguir apresenta o resultado do teste.



Fig. 9. Resultado do teste do uso do modelo YOLO. Fonte: Autores

3) Teste do Telegram: O teste do Telegram teve como principal função verificar a possibilidade da comunicação entre a Raspberry Pi com usuários do aplicativo Telegram, sendo o objetivo principal da comunicação o envio de mensagens e fotos. O teste foi desenvolvido em Python 3, por ser a primeira vez da equipe utilizado a API Bot do Telegram. A figura 10 a seguir apresenta o resultado do teste.

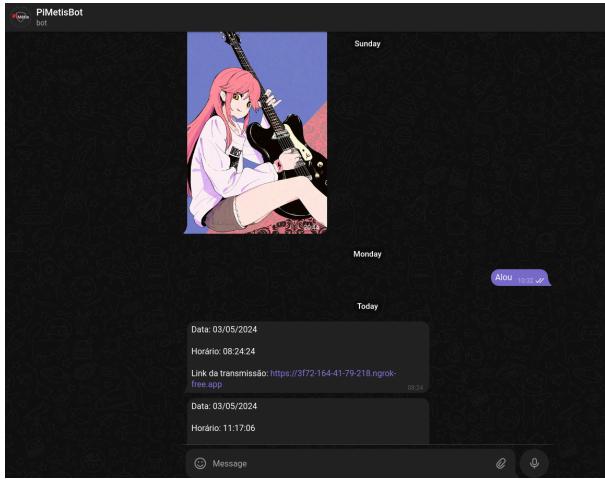


Fig. 10. Resultado do teste de envio de mensagens por bot do Telegram. Fonte: Autores

4) Teste do Flask e ngrok: O teste do uso Flask e do ngrok teve como principal função verificar se é possível criar uma página web para transmitir ao vivo pela internet as imagens da câmera. Foi utilizado o Python 3 no desenvolvimento deste teste pela sua facilidade de criar serviços web integrando com o OpenCv para aquisição de imagens da webcam, junto com a sua biblioteca do ngrok para ser possível a transmissão de um link pela internet.

A figura 11 a seguir apresenta o resultado teste.



Fig. 11. Resultado do teste de transmissão ao vivo pela internet das imagens da webcam. Fonte: Autores

B. Ponto de Controle II: C++

Após ter sido realizados os testes dos subsistemas em uma linguagem de programação de alto nível como o Python pela facilidade e velocidade do desenvolvimento, foi feito a passagem dos subsistemas para a linguagem C++ pelo seu desempenho e velocidade, pelo sistema ser executado em um ambiente com poucos recursos computacionais.

1) Subsistema da Interface Gráfica: O teste realizado da interface gráfica inicialmente havia sido realizado em C++, portanto, o seu resultado pode ser lido em IX-A1. **Recursos básicos do sistema:** Neste subsistema, foram utilizados dois recursos do sistema operacional. Primeiramente, foram empregadas threads para criar uma execução paralela da captação de imagens da câmera e da execução da interface gráfica. Em segundo lugar, foram utilizados mutexes para proteger o acesso simultâneo à variável que armazena o frame da câmera, garantindo que o OpenCV possa atualizar o frame enquanto a interface gráfica atualiza o widget correspondente de forma segura.

2) Teste do YOLO: O teste realizado do YOLO foi passado para a linguagem C++ com sucesso, pelo grande suporte a linguagem no framework, além dessa conversão de linguagem, também foi feito uma alteração no algoritmo para ignorar o restante das classes que não são o objetivo deste sistema. **Recursos básicos do sistema:** Neste subsistema, foram utilizados dois recursos do sistema operacional. Primeiramente, sinais foram empregados para atualizar variáveis de contagem, permitindo a comunicação entre diferentes threads ou processos. Em segundo lugar, para garantir que a detecção de pessoas ou armas não fosse duplicada, foram utilizados temporizadores que asseguram um intervalo seguro entre as detecções consecutivas. É possível realizar a observação desse resultado pela figura 12.

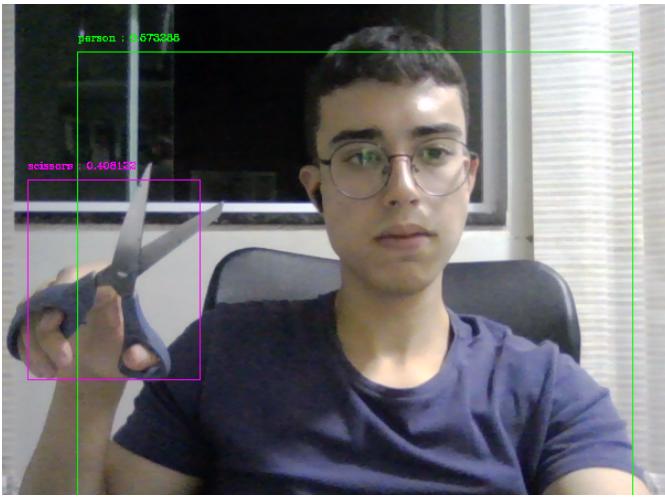


Fig. 12. Segundo resultado do teste do uso do modelo YOLO. Fonte: Autores

3) *Subsistema do Telegram:* O teste do Telegram foi passado para a linguagem C++ com sucesso, criando uma mini biblioteca enxuta de bot de Telegram, na qual foi necessário utilizar bibliotecas externas de JSON Nlohman^[13] e de requisições HTTP^[14]. **Recursos básicos do sistema:** Neste subsistema, foi utilizado um recurso do sistema operacional, sendo thread para criar uma segunda linha de execução que ficará escutando o chat, para assim acionar o algoritmo correspondente do comando enviado pelo usuário, como apresentado na figura 13.

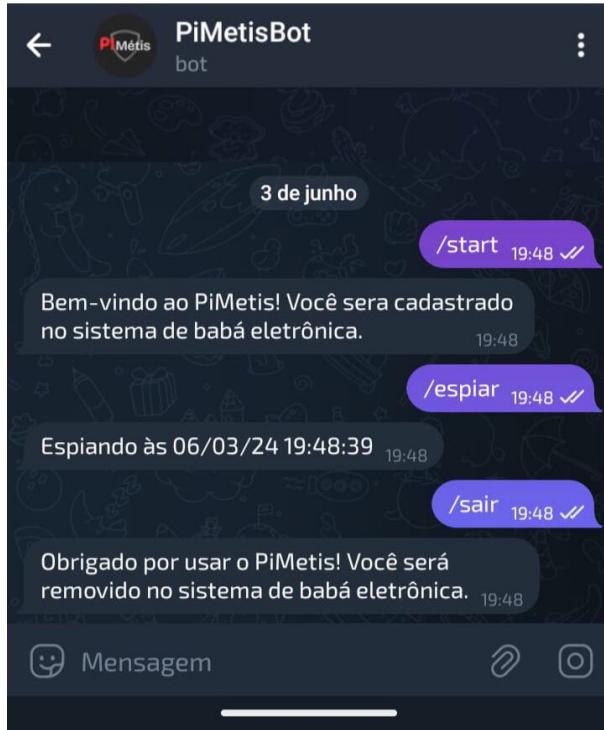


Fig. 13. Segundo resultado do teste do uso do telegram. Fonte: Autores

4) *Subsistema da Transmissão ao vivo:* O teste do Flask e Ngrok não foi possível ser passado para C++ por enquanto, por questões de dificuldades na troca de linguagem, na qual em C++ fica mais complexo, necessitando de mais tempo para o seu desenvolvimento. **Recursos básicos do sistema:** Neste subsistema, foi utilizado um recurso do sistema operacional, sendo thread para criar uma segunda linha de execução que irá criar e manter um servidor ngrok ao mesmo tempo que é executado um servidor http local.

C. Ponto de Controle III

Feito a passagem dos subsistemas para a linguagem de programação C++ e utilizando recursos do sistema operacional, foi feito a integração entre os susbsistemas, onde foi ligada a interface gráfica para visualização ao vivo da câmera e a configuração do limite de pessoas para o sistema notificar que pode ser visto na figura 14, e a integração entre o YOLO com o bot de Telegram onde o resultado pode ser visto na figura 15.

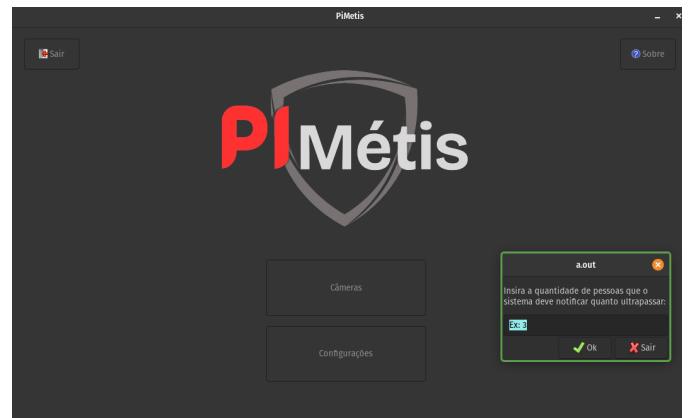


Fig. 14. Integração GUI e YOLO. Fonte: Autores

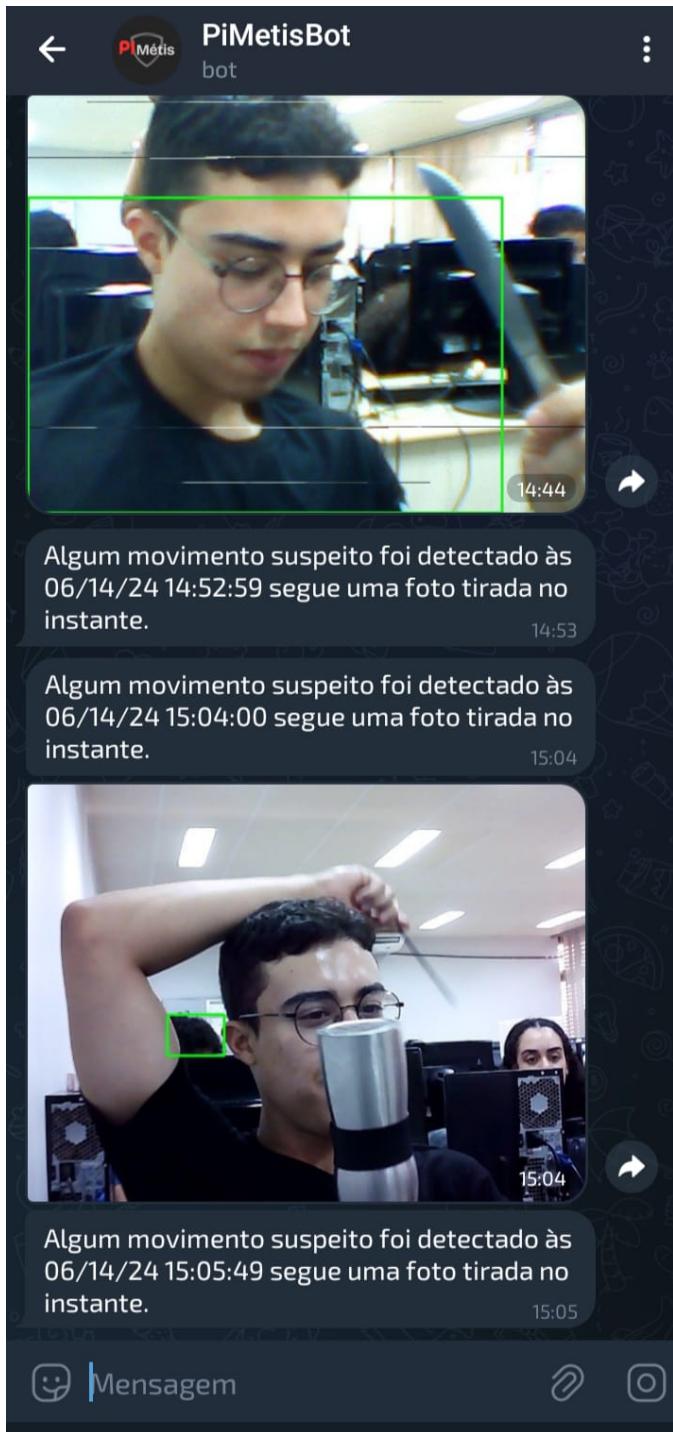


Fig. 15. Integração bot do Telegram e YOLO. Fonte: Autores

X. BIBLIOGRAFIA

- [1] AXIS Camera Station Secure Entry
(Disponível em: <https://www.axis.com/pt-br/products/axis-camera-station-secure-entry-support-and-resources> . Acesso em: 02 de abril de 2024)
- [2] Videovigilância inteligente da QNAP

(Disponível em: <https://www.qnap.com/solution/qvr-surveillance/pt-br/> . Acesso em: 02 de abril de 2024)

[3] HP7 2K Interfone com vídeo residencial inteligente
(Disponível em: <https://www.ezviz.com/br/product/HP7/50223> . Acesso em: 02 de abril de 2024)

[4] Alarmes Verisure: Sistemas de Monitoramento
(Disponível em: <https://www.verisure.com.br/> . Acesso em: 02 de abril de 2024)

[5] C++ Interfaces for GTK and GNOME
(Disponível em: <https://gtkmm.org/en/index.html> . Acesso em: 29 de abril de 2024)

[6] YOLO Algorithm: Real-Time Object Detection from A to ZE

(Disponível em: <https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z-yolo-algorithm-how-does-it-works?> . Acesso em: 30 de abril de 2024)

[7] darknet: YOLOv4

(Disponível em: <https://github.com/AlexeyAB/darknet> . Acesso em: 29 de abril de 2024)

[8] Telegram Bot API

(Disponível em: <https://core.telegram.org/bots/api> . Acesso em: 30 de abril de 2024)

[9] OpenCV: Cascade Classifier

(Disponível em: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html . Acesso em: 30 de abril de 2024)

[10] Haar Cascades, Explained

(Disponível em: <https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d> . Acesso em: 30 de abril de 2024)

[11] Camera App with Flask and OpenCV

(Disponível em: <https://towardsdatascience.com/camera-app-with-flask-and-opencv-bd147f6c0eec> . Acesso em: 28 de abril de 2024)

[12] Python — ngrok documentation

(Disponível em: <https://ngrok.com/docs/using-ngrok-with/python/> . Acesso em: 30 de abril de 2024)

[13] JSON for Modern C++

(Disponível em: <https://json.nlohmann.me/> . Acesso em: 25 de maio de 2024)

[14] cpr - C++ Requests

(Disponível em: <https://docs.libcpr.org/> . Acesso em: 25 de maio de 2024)