

**ESTÁCIO DE SÁ
NOVA AMÉRICA**

Snap Link

**Leonardo Machado Molinaro
Lucas Antunes Floriano**

**2024
Rio de Janeiro/Rio de Janeiro**

Sumário

1. DIAGNÓSTICO E TEORIZAÇÃO	3
1.1. Identificação das partes interessadas e parceiros	3
1.2. Problemática e/ou problemas identificados	6
1.3. Justificativa	6
1.4. Objetivos/resultados/efeitos a serem alcançados (em relação ao problema identificado e sob a perspectiva dos públicos envolvidos)	7
1.5. Referencial teórico (subsídio teórico para propositura de ações da extensão)	7
2. PLANEJAMENTO E DESENVOLVIMENTO DO PROJETO	9
2.1. Plano de trabalho (usando ferramenta acordada com o docente)	9
2.2. Descrição da forma de envolvimento do público participante na formulação do projeto, seu desenvolvimento e avaliação, bem como as estratégias	9
2.3. Grupo de trabalho (descrição da responsabilidade de cada membro)	10
2.4. Metas, critérios ou indicadores de avaliação do projeto	11
2.5. Recursos previstos	12
2.6. Detalhamento técnico do projeto	12
3. ENCERRAMENTO DO PROJETO	14
3.1. Relato Coletivo:	14
3.1.1. Avaliação de reação da parte interessada	15
3.2. Relato de Experiência Individual	15
3.2.1. CONTEXTUALIZAÇÃO	15
3.2.2. METODOLOGIA	16
3.2.3. RESULTADOS E DISCUSSÃO:	16
3.2.4. REFLEXÃO APROFUNDADA	22
3.2.5. CONSIDERAÇÕES FINAIS	22

1. DIAGNÓSTICO E TEORIZAÇÃO

1.1. Identificação das partes interessadas e parceiros

No projeto, Leonardo Molinaro, graduando em Ciência da Computação, assume a responsabilidade de criar uma solução tecnológica inovadora. Seu objetivo é aplicar na prática as habilidades adquiridas ao longo de sua formação, desenvolvendo um sistema que soluciona a problemática da parte interessada. Essa experiência permite que Leonardo não só refine suas competências técnicas, mas também interaja com um cliente real, compreendendo e atendendo às necessidades específicas da parte interessada.

Bianca Moraes, a parte interessada, é uma fisioterapeuta graduada e exerce sua profissão em uma clínica, e está constantemente em busca de soluções que possam melhorar a eficiência no ambiente em que trabalha.

Para garantir que todas as expectativas e responsabilidades estejam claramente definidas, foi estabelecido um Termo de Acordo de Cooperação entre as duas partes. Esse acordo formaliza a colaboração e reforça o compromisso de Leonardo em entregar uma solução eficaz e adaptada às necessidades de Bianca.

 Espaço de Acupuntura e Fisioterapia Especializada

CARTA DE APRESENTAÇÃO

Vimos por esta apresentar o grupo de acadêmicos da Universidade Estácio de Sá – UNESA listado na tabela a final deste documento, a fim de convidá-lo a participar de uma atividade extensionista associada à disciplina PROGRAMAÇÃO DE MICROCONTROLADORES sob responsabilidade do Prof. Lucas Antunes Floriano.

Em consonância ao Plano Nacional de Educação e demais normativas educacionais vigentes, a Universidade Estácio de Sá – UNESA desenvolve atividade extensionista que, norteados pela metodologia de aprendizagem baseada em projetos, tem por princípios fundantes o diagnóstico dos problemas/demandas/necessidades, a participação ativa dos interessados/públicos participantes, a construção dialógica, coletiva e experencial de conhecimentos, o planejamento de ações, o desenvolvimento e avaliação das ações, a sistematização dos conhecimentos, a avaliação das ações desenvolvidas.

Nesse contexto, a disciplina acima mencionada tem como principal escopo os temas relacionados à aplicação de microcontroladores, projetos de prototipagem e comunicação de microcontroladores com sensores e atuadores.

Sendo assim, pedimos o apoio dessa organização/entidade/coletivo/associação/outro, que aqui chamaremos de parte interessada, para a realização das seguintes atividades: diagnósticos, análises, entrevistas, levantamentos, projetos ou qualquer outra metodologia de estudo de caso que auxiliem no desenvolvimento das competências de nossos acadêmicos e ao mesmo tempo possa contribuir para a comunidade em que estamos inseridos.

Como se trata de atividade de ensino/aprendizagem de caráter extensionista, prevista no Projeto Pedagógico do Curso, salientamos que:

- não há cobrança de remuneração de qualquer natureza por parte da Universidade Estácio de Sá, seus alunos ou o docente da disciplina, à parte interessada;
- as atividades desenvolvidas no âmbito do projeto extensionista não configuram relação de trabalho entre os alunos e o docente da Universidade Estácio de Sá – UNESA disciplina PROGRAMAÇÃO DE MICROCONTROLADORES, e a parte interessada;
- os resultados do projeto só poderão ser implantados para uso efetivo mediante Anotação de Responsabilidade Técnica de um profissional habilitado;
- os resultados do projeto podem ser implantados pela parte interessada para fins lucrativos, sem a necessidade de pagamento de quaisquer benefícios aos alunos, ao docente da disciplina e à Universidade Estácio de Sá – UNESA;
- quaisquer custos relativos à implantação e operação contínua do projeto fora do escopo das atividades do presente projeto serão arcados pela parte interessada.

Aproveitamos a oportunidade e solicitamos que, em caso de aceite, seja formalizado, mediante assinatura da Carta de Autorização, as atividades e informações que o(s) aluno(s) poderá(ão) ter acesso.

Desde já nos colocamos à sua disposição para quaisquer esclarecimentos. Professor Lucas Antunes Floriano – 98820-4968 e/ou lucas.floriano@estacio.br e Leonardo Machado Molinaro - leomolinarodev@gmail.com e/ou <https://github.com/LMolinaro01>

Grupo de Alunos
Leonardo Machado Molinaro Matrícula: 202302954928

Atenciosamente,

Lucas Antunes

Lucas Antunes Floriano

Docente da disciplina: PROGRAMAÇÃO DE MICROCONTROLADORES

Semestre: 2024.2

Matrícula: 10643417

Bianca Moraes
Bianca Cordeiro de Moraes
CPF: 17269247773

Rio de Janeiro, 18 de Agosto de 2024.

1.2. Problemática e/ou problemas identificados

A gestão eficaz de uma clínica de fisioterapia depende de um controle rigoroso sobre as movimentações de pacientes e colaboradores. No entanto, a ausência de um sistema automatizado de monitoramento pode gerar uma série de problemas que afetam diretamente a operação diária e a segurança do ambiente clínico. Um dos principais desafios é a dificuldade de acompanhar as entradas e saídas no consultório, o que pode levar a falhas na organização do fluxo de pessoas. Essa falta de controle pode resultar em atrasos nos atendimentos, comprometendo a pontualidade e a satisfação dos pacientes, além de dificultar a gestão eficiente do espaço.

Além disso, a ausência de um sistema de monitoramento deixa brechas na segurança do local, uma vez que atividades suspeitas ou entradas não autorizadas podem passar despercebidas. Em clínicas movimentadas, é essencial que haja um registro preciso e contínuo das movimentações para garantir um ambiente seguro para todos os presentes.

Por fim, sem um mecanismo automatizado para alertar sobre o fluxo de entrada e saída, a equipe precisa dedicar mais tempo e recursos para tarefas de vigilância, que poderiam ser melhor empregadas em outras atividades importantes para o funcionamento e a qualidade do atendimento na clínica.

1.3. Justificativa

A implementação de uma câmera de segurança integrada com Python, capaz de registrar entradas e saídas e armazenar automaticamente esses dados no computador, representa um avanço significativo na gestão e organização de clínicas. O projeto visa facilitar o registro de informações essenciais, como horários de chegada e saída dos pacientes, tornando o acompanhamento mais preciso e acessível. Esse controle otimizado de horários permite uma melhor organização logística dos atendimentos, ajudando a reduzir atrasos e a garantir que a agenda dos profissionais seja seguida de forma mais eficiente.

Além da otimização do fluxo de trabalho, o sistema também adiciona uma camada de segurança ao ambiente clínico, monitorando continuamente a movimentação e fornecendo uma visão clara de quem entra e sai do espaço. Isso não só ajuda a prevenir situações indesejadas, como também cria um ambiente mais seguro e tranquilo para pacientes e funcionários. Dessa forma, a tranquilidade e confiança de que há um sistema ativo monitorando o ambiente contribuem para uma experiência mais positiva e segura para todos.

O projeto, portanto, não se limita apenas à automação de processos administrativos, mas também visa criar uma solução integrada que melhore a eficiência clínica e garanta a

segurança. Ao coletar e armazenar dados de forma automática, ele promove uma gestão mais organizada, economizando tempo e recursos que podem ser redirecionados para melhorar a qualidade do atendimento. Com isso, tanto os profissionais de saúde quanto os pacientes desfrutam de uma experiência mais fluida e eficiente no ambiente clínico.

1.4. Objetivos/resultados/efeitos a serem alcançados (em relação ao problema identificado e sob a perspectiva dos públicos envolvidos)

O projeto tem como principal objetivo desenvolver uma solução integrada para captura e monitoramento de imagens, oferecendo uma ferramenta eficiente para a gestão de segurança e organização em ambientes clínicos. A aplicação permite a captura automática de imagens, que são salvas com carimbo de data e hora, facilitando o controle de registros importantes, como horários de movimentação. Dessa forma, essa funcionalidade visa proporcionar um recurso que otimize o gerenciamento do espaço, reduzindo a necessidade de supervisão manual e aumentando a eficiência nas operações cotidianas.

Outro objetivo essencial é a criação de uma interface gráfica intuitiva e amigável, que simplifique a interação dos usuários com o sistema. Tal qual outras soluções modernas, a interface foi projetada para facilitar a captura, visualização e agendamento de imagens, além de oferecer feedback em tempo real sobre as operações realizadas. Por conseguinte, isso garante que os usuários possam operar o sistema de forma ágil e sem complicações, melhorando a experiência e a produtividade ao usar a aplicação no dia a dia.

Para garantir o pleno funcionamento da aplicação, é fundamental assegurar uma comunicação estável e contínua entre a aplicação Python e os dispositivos Arduino e ESP32. A fim de alcançar essa estabilidade, a sincronização automática entre esses componentes permite que a captura de imagens e a troca de dados ocorram de maneira fluida, sem interrupções, oferecendo uma solução robusta e confiável para monitoramento e coleta de informações. Visto que a solidez dessa comunicação é essencial para o sucesso do sistema, é crucial que ele possa operar em diferentes contextos de forma prática e eficiente, garantindo sua eficácia.

1.5. Referencial teórico (subsídio teórico para propositura de ações da extensão)

É importante destacar, em primeiro plano, que a integração de tecnologias como IoT, ESP32 e Python em sistemas de monitoramento tem se mostrado uma solução eficaz tanto para vigilância quanto para o gerenciamento clínico. Conforme Sharma e Gupta (2018)

demonstram, o uso de IoT em sistemas de vigilância permite a automação eficiente da coleta de dados e a resposta rápida a eventos. O projeto em desenvolvimento, que utiliza o ESP32-CAM, segue essa linha de otimização ao capturar e gerenciar imagens automaticamente, proporcionando uma abordagem mais conectada e prática para segurança e gestão clínica. A funcionalidade de captura de imagens, combinada com recursos como salvamento automático com carimbo de data e hora, oferece um controle preciso sobre os registros, otimizando processos e reduzindo a necessidade de intervenção manual.

Outrossim, a pesquisa de Rai e Rehman (2019) reforça o potencial do ESP32 em aplicações de monitoramento em tempo real, destacando sua capacidade de capturar eventos de maneira automatizada. No projeto, essa funcionalidade é essencial, validando a escolha do ESP32 como a plataforma central para desenvolver um sistema acessível e eficiente de vigilância, que também pode ser adaptado ao ambiente clínico para monitoramento detalhado. Essa flexibilidade no uso do ESP32 ressalta como uma tecnologia de baixo custo pode trazer grandes benefícios tanto para a segurança quanto para a gestão em saúde, oferecendo um meio de registrar e acessar dados com praticidade e precisão.

Além disso, a implementação de funcionalidades como agendamento de capturas e a definição de temporizadores encontra paralelos claros no sistema que estou desenvolvendo, mostrando a versatilidade do microcontrolador ESP32 em lidar com diferentes formas de comunicação e resposta a eventos. A capacidade de definir intervalos automáticos e agendar capturas futuras permite uma coleta de dados mais organizada e contínua, algo essencial para ambientes que exigem monitoramento constante. Dessa forma, a arquitetura proposta demonstra uma eficiência robusta em soluções de monitoramento que visam otimizar processos e maximizar recursos, mantendo um fluxo de operações simplificado e intuitivo.

Por fim, o livro de Pratik Desai, "Python Programming for Arduino," é uma base importante para compreender a integração entre Python e Arduino, essencial para ampliar as funcionalidades do projeto. Desai explora como o Python pode interagir com hardware de forma simples e eficiente, facilitando a programação e a automação de tarefas como a captura e salvamento de imagens. Essa abordagem reflete diretamente no sistema, que busca utilizar o ESP32-CAM com máxima eficiência para proporcionar um monitoramento automatizado e organizado, com uma interface gráfica que oferece uma experiência de usuário simplificada e direta.

Por conseguinte, ao alinhar as tecnologias exploradas nesses estudos e na literatura técnica com o desenvolvimento atual, o *Snap Link* não só integra as melhores práticas de automação e monitoramento, mas também se posiciona como uma solução que pode ser aplicada em diferentes contextos, desde a segurança até o gerenciamento clínico. Isso representa uma oportunidade de transformar como monitoramos e gerenciamos ambientes

críticos, com a ajuda de tecnologias emergentes que otimizam a coleta e o acesso a informações relevantes.

[1] Sharma, M., & Gupta, S. C. (2018, June). *An internet of things based smart surveillance and monitoring system using Arduino*. In 2018 International Conference on Advances in Computing and Communication Engineering (IICACCE) (pp. 428-433). IEEE.

[2] Rai, P., & Rehman, M. (2019, January). *ESP32 based smart surveillance system*. In 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (pp. 1-3). IEEE.

[3] Desai, Pratik. *Python Programming for Arduino*. Packt Publishing Ltd, 2015.

2. PLANEJAMENTO E DESENVOLVIMENTO DO PROJETO

2.1. Plano de trabalho (usando ferramenta acordada com o docente)

The image shows a digital project management board with three main sections:

- Contato com o Cliente**: Contains three items: "Primeiro Contato (E-mail)" (284 words), "Segundo Contato (Formulário)" (30 words), and "Terceiro Contato (Presencial)" (27 words). A link to a Google Doc is provided: <https://docs.google.com/document/d/1nLxqK8Roteiro-de-Extensao>.
- Lista de Tarefas - Leonardo (Único Integrante)**: A detailed list of tasks with checkboxes:
 - Círculo Elétrico
 - Escolher todos os componentes
 - Criar um Protótipo Visual
 - Realizar a Montagem
 - Testar o ESP32
 - Criação de um server Remoto
 - Testes usando serial print
 - Integrar o Arduino com Python
 - Envio da imagem base 64 via Serial
 - Comunicação Python -> ESP32/Arduin
 - Comunicação Serial
 - Python executa o comando de foto e recebe a base64
 - Base64 -> Imagem
 - Criação de uma GUI
 - Planejar no Quadro branco
 - Escolher uma ferramenta
 - Custom Tkinter
 - Definir o Número de Telas
 - Funções de tirar foto
 - Captura única
 - Temporizador
 - Agendamento
 - Tratamento de Exceções e Validações
 - Testes Automatizados Unitários
- LMolinaro01/SnapLink**: A GitHub repository for a "Projeto de câmera de segurança inteligente que combina Arduino, Python e ESP32 para oferecer uma solução acessível e eficaz para..." with 1 contributor, 0 issues, 0 stars, and 0 forks. The URL is <https://github.com/LMolinaro01/SnapLink>. It includes a "Repositório do Projeto" button.
- Frontend**: Shows a whiteboard diagram illustrating the system architecture and data flow between Arduino, Python, and ESP32. Below it is a "Protótipo Visual do Sistema" section with 6 cards.

2.2. Descrição da forma de envolvimento do público participante na formulação do projeto, seu desenvolvimento e avaliação, bem como as estratégias

No desenvolvimento do projeto, a participação dos atores sociocomunitários foi

essencial e ocorreu de diversas formas. Inicialmente, estabeleci contato através de e-mails, onde apresentei a proposta do projeto e convidei-a para participar. Esse primeiro passo foi fundamental para abrir um canal de comunicação e envolver todos os interessados.

Para capturar as preferências e necessidades da comunidade, criei um formulário detalhado, que foi distribuído e preenchido pelo cliente. Esse formulário me forneceu informações valiosas sobre as expectativas e sugestões.

Além disso houve um encontro presencial para exibir as funcionalidades do programa e orientar o cliente como utilizá-lo



2.3. Grupo de trabalho (descrição da responsabilidade de cada membro)

O projeto *Snap Link* está sendo conduzido exclusivamente por Leonardo Machado Molinaro, que assume diversas responsabilidades para garantir seu desenvolvimento eficaz. Ele é responsável por:

Desenvolvimento do código fonte: Leonardo cuida da programação e implementação do código fonte do sistema, utilizando Arduino, Python e ESP32. Isso inclui a escrita, depuração e otimização do código para assegurar que o sistema funcione corretamente e atenda às necessidades de monitoramento.

Montagem eletrônica: Além do desenvolvimento do código, Leonardo também é encarregado da montagem eletrônica do dispositivo, garantindo que todos os componentes, como a câmera e os sensores, estejam corretamente integrados e funcionando em conjunto.

Comunicação e interação com o cliente: Leonardo mantém uma comunicação contínua com o cliente, compreendendo seus requisitos e coletando feedback ao longo do processo. Para facilitar essa interação, ele utiliza ferramentas como e-mail e reuniões presenciais, permitindo uma troca de informações clara e produtiva.

Elaboração do roteiro de extensão e carta de apresentação: Leonardo é responsável por criar um roteiro abrangente para o projeto, delineando recursos futuros a serem implementados e prazos necessários, além de elaborar a carta de apresentação, que contextualiza o projeto e sua relevância.

2.4. Metas, critérios ou indicadores de avaliação do projeto

Para atingir os objetivos estabelecidos no projeto, é imprescindível um detalhamento cuidadoso das etapas que serão seguidas, uma vez que isso permitirá uma execução eficaz e organizada do plano de ação. Essas fases abrangem desde a definição das funcionalidades da aplicação até a implementação e a avaliação do sistema em sua totalidade. Dessa forma, apresento a seguir um plano que descreve como os objetivos serão alcançados, além de delinear os critérios e indicadores necessários para medir a efetividade da solução.

Em primeiro lugar, para desenvolver uma solução integrada de captura e monitoramento de imagens, a etapa inicial consiste na definição minuciosa das funcionalidades principais do sistema. Isso abrange a captura automática de imagens e o salvamento local com carimbo de data e hora. Essa definição clara e precisa é essencial, pois orientará todas as fases subsequentes do projeto, garantindo que cada aspecto da aplicação atenda às necessidades específicas dos usuários.

Na sequência, a segunda etapa é dedicada ao desenvolvimento da interface gráfica, onde se busca criar um ambiente amigável e intuitivo para o usuário. Nessa fase, serão realizados testes de usabilidade para assegurar que a interface não apenas facilite a captura e visualização de imagens, mas também permite o agendamento de forma descomplicada. Os critérios para esta etapa incluem: a clareza dos controles, a facilidade de navegação e a

eficiência da interação do usuário com o software. Assim sendo, o feedback obtido durante os testes será fundamental para refinar a interface e garantir uma experiência positiva.

Além disso, a terceira etapa envolve a implementação da comunicação entre a aplicação em Python e os dispositivos Arduino e ESP32. Esta fase é crucial, pois envolve a programação e os testes de conexão automática, assegurando que a sincronização entre os componentes funcione sem falhas. A fim de garantir a estabilidade e a fluidez do sistema, serão definidos critérios específicos para a comunicação, como a latência na troca de dados e a taxa de erro de conexão.

Por conseguinte, essas etapas, alinhadas aos critérios e indicadores de sucesso, formarão uma base sólida para a efetividade da solução proposta. Ao adotar uma abordagem sistemática e focada na usabilidade e integração, o projeto não apenas poderá alcançar seus objetivos, mas também proporcionará uma ferramenta valiosa para a gestão de segurança e organização em ambientes clínicos. Nesse sentido, a implementação bem-sucedida das fases delineadas garantirá que a aplicação atenda às expectativas e se destaque como uma solução inovadora e eficiente.

2.5. Recursos previstos

Recursos Materiais:

Computadores e Software: O projeto utilizará a máquina pessoal do desenvolvedor, equipada com o ambiente de desenvolvimento PyCharm, Arduino IDE e as seguintes bibliotecas: *"serial, time, base64, tkinter, PIL (Pillow), custom tkinter, io, datetime, threading, os, esp_camera.h, Arduino.h, base64.h"*.

Hardware: Placa Arduino UNO R3 + Cabo USB tipo B, Placa ESP32 CAM com Câmera OV2640 2MP, 6 jumpers macho-macho, 1 jumper fêmea-fêmea, Extensor USB.

Custo Total: R\$ 144,20 (Outubro de 2024, Frete incluso)

Recursos Humanos:

Desenvolvedor: Leonardo, responsável pelo desenvolvimento do Sistema, montagem eletrônica, comunicação e interação com o cliente e toda a elaboração do roteiro de extensão e carta de apresentação

Cliente: Responsável por fornecer requisitos, feedback e interações.

2.6. Detalhamento técnico do projeto

O *Snap Link* é uma aplicação em Python que integra a comunicação entre um Arduino e um ESP32, permitindo a captura, visualização e armazenamento de imagens com carimbo de data e hora. Desenvolvido com a biblioteca *Custom Tkinter*, o projeto oferece uma interface gráfica moderna e intuitiva, que proporciona um controle eficiente das funcionalidades. Seu foco é automatizar a captura e gerenciamento de imagens, garantindo uma comunicação precisa entre hardware e software.

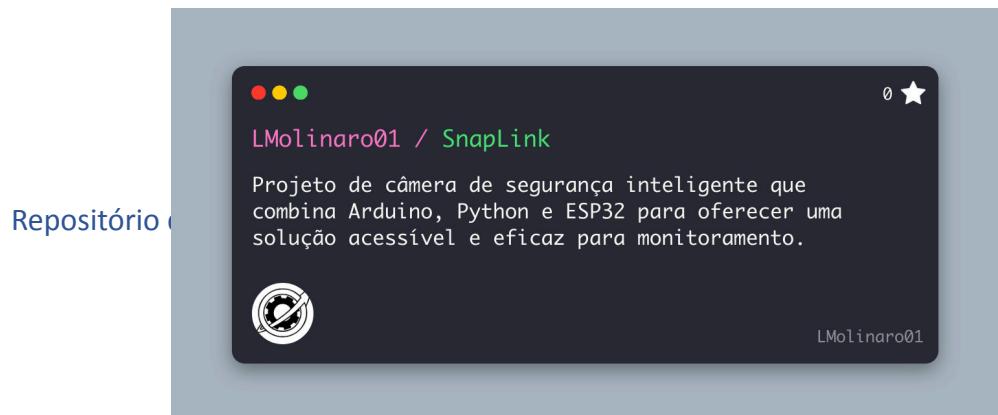
A conexão entre a aplicação e o Arduino é feita via comunicação serial, utilizando a biblioteca serial. Ao iniciar o programa, ele tenta estabelecer automaticamente a comunicação. Em caso de falha, o sistema oferece ao usuário a opção de tentar novamente por meio de uma janela interativa. A captura das imagens é realizada pelo ESP32, que envia os dados em formato *base64*. A aplicação decodifica esses dados com a biblioteca *base64*, exibindo as imagens diretamente na interface, sem a necessidade de abrir novas janelas.

As imagens capturadas podem ser salvas nos formatos JPEG ou PNG. Para maior precisão, a data e hora da captura são impressas automaticamente no canto inferior direito, utilizando o módulo *datetime* e funcionalidades da biblioteca *Pillow*. O sistema também permite que o usuário defina temporizadores para capturas automáticas, em intervalos configuráveis de segundos, minutos ou horas. Caso o usuário prefira, é possível agendar capturas para horários futuros, garantindo flexibilidade no gerenciamento das imagens.

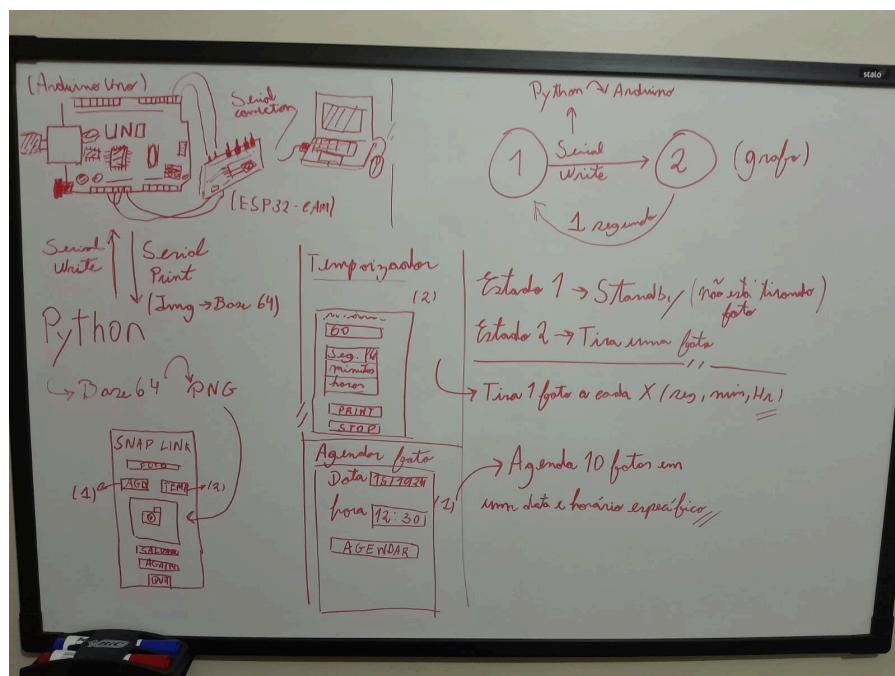
A fim de manter a interface responsiva e evitar bloqueios durante a execução, a aplicação utiliza threads para operações simultâneas, como a captura e o salvamento das imagens. Por exemplo, ao programar capturas contínuas, cada imagem é salva automaticamente em uma pasta selecionada, com nomes baseados em timestamps, garantindo organização eficiente. A aplicação possui também uma funcionalidade de agendamento com uma contagem regressiva visual, atualizada em tempo real, para que o usuário saiba exatamente quando cada captura será realizada.

O código foi cuidadosamente estruturado para lidar com falhas de conexão e oferecer uma experiência estável. Testes unitários, desenvolvidos com a biblioteca *unittest*, validam a comunicação com o Arduino e garantem que todos os botões e funcionalidades da interface operem corretamente. O comando “*python -m unittest discover*” pode ser utilizado para executar esses testes, assegurando que cada componente funcione como esperado.

O *Snap Link* é, portanto, uma solução prática e robusta que une hardware e software em perfeita sintonia. A aplicação é ideal para projetos que requerem automação na captura de imagens, oferecendo ao usuário uma interface responsiva, ferramentas avançadas de agendamento e uma integração confiável com microcontroladores.



Grafo e Planejamento Geral:



3. ENCERRAMENTO DO PROJETO

3.1. Relato Coletivo:

Um dos principais desafios enfrentados durante o desenvolvimento do projeto foi a necessidade de lidar com a parte eletrônica, algo que nunca havia feito antes. Essa experiência me proporcionou aprendizados valiosos em termos acadêmicos.

Destaca-se também que todo o trabalho foi realizado de forma independente, o que exigiu uma grande dose de autodisciplina, organização e habilidades técnicas.

Em suma, o sistema foi bem-sucedido em atingir os objetivos sociocomunitários propostos, contribuindo para uma prática mais eficiente dos profissionais de saúde. O trabalho realizado de forma independente e a experiência adquirida no trato com o cliente

acrescentaram ainda mais valor ao projeto, consolidando-o como uma solução relevante e impactante para a comunidade.

3.1.1. Avaliação de reação da parte interessada

Pesquisa de Satisfação

Como você classificaria sua satisfação geral com o Projeto?

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Quão provável é que você recomende o Sistema a outras pessoas?

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

O quão fácil foi utilizar a Aplicação?

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Você encontrou algum problema ao usar o Sistema? Se sim, por favor, descreva.

Não encontrei problema algum.

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários

3.2. Relato de Experiência Individual

3.2.1. CONTEXTUALIZAÇÃO

Desenvolvi o projeto de forma autônoma, abrangendo desde a concepção até a implementação do sistema. Fui responsável por todas as etapas do processo, com o objetivo principal de solucionar a problemática da captura e gerenciamento de imagens em tempo real. Este projeto tecnológico visa otimizar a comunicação entre um Arduino e um ESP32, permitindo que profissionais de saúde possam capturar, visualizar e armazenar imagens de forma eficiente, melhorando a experiência de tratamento dos pacientes.

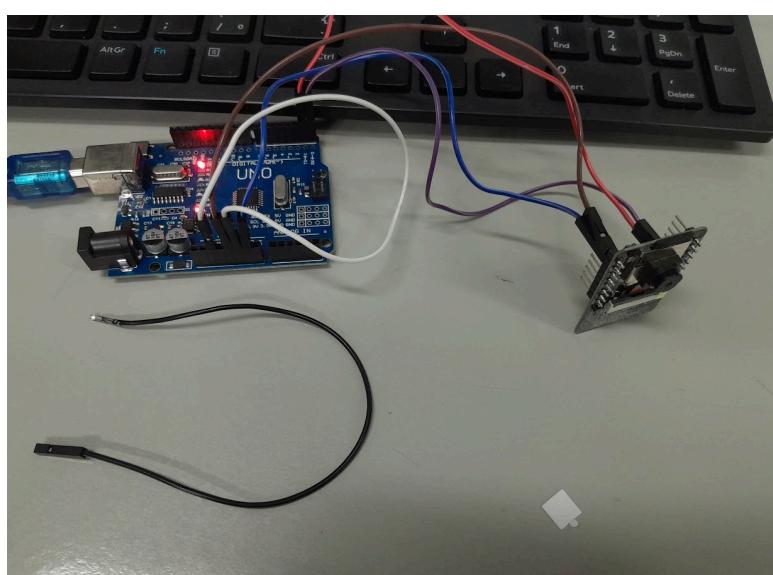
3.2.2. METODOLOGIA

A experiência no desenvolvimento do projeto envolveu interações regulares com a parte interessada para compreender suas necessidades e expectativas. Ao longo de vários meses, passei por um processo de planejamento detalhado, que incluiu a análise de requisitos, a definição da arquitetura do sistema e a elaboração de um cronograma de desenvolvimento. Embora tenha trabalhado de forma independente na execução do projeto, muitas das decisões foram validadas com a parte interessada para assegurar que estivessem alinhadas com suas demandas, garantindo que a solução final atendesse eficazmente às necessidades de captura e gerenciamento de imagens em tempo real.

3.2.3. RESULTADOS E DISCUSSÃO:

O desenvolvimento inicial foi marcado por uma certa incerteza sobre o que criar, uma vez que a programação de microcontroladores com Arduino oferece uma vasta gama de possibilidades criativas. Com base na problemática citada anteriormente, resolvi direcionar meus esforços para construir um sistema de segurança que integrasse o ESP32 e o Arduino. No entanto, ainda não havia formado uma ideia clara de como o sistema funcionaria. Sabia que utilizaria Python para conectar os dois dispositivos e já havia planejado capturar movimentos com um sensor ultrassônico, enviando uma foto via WhatsApp. Contudo, logo percebi que essa ideia, além de simplista, era algo já bastante disseminado, considerando que câmeras de segurança com sensor de movimento existem há anos no mercado.

Com a decisão tomada, adquiri os materiais necessários e iniciei os testes com a câmera ESP32. De início, enfrentei dificuldades consideráveis, especialmente na transferência do código da IDE Arduino para o ESP32. Um obstáculo foi a necessidade de usar um cabo fêmea-fêmea, que precisei improvisar removendo um pino de um jumper macho-fêmea. A configuração da placa também trouxe desafios, tornando a simples tarefa de passar o código para o ESP32 um processo árduo e sujeito a falhas devido ao cabo improvisado.



```

CameraWebServer.ino app_httpd.cpp camera_index.h camera_pins.h ci.json
31 // #define CAMERA_MODEL_ESP32S3_CAM_LCD
32 // #define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
33 // #define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
34 #include "camera_pins.h"
35
36 // =====
37 // Enter your WiFi credentials
38 // =====
39 const char *ssid = "MOLINARIO_5G";
40 const char *password = "Casamento0205";
41
42 void startCameraServer();
43 void setupLedFlash(int pin);
44
45 void setup() {
46   Serial.begin(115200);
47   Serial.setDebugOutput(true);
48   Serial.println();
49
50   camera_config_t config;
51   config.ledc_channel = LEDC_CHANNEL_0;
52   config.ledc_timer = LEDC_TIMER_0;
53   config.pin_d0 = Y2_GPIO_NUM;
54   config.pin_d1 = Y3_GPIO_NUM;
55   config.pin_d2 = Y4_GPIO_NUM;

```

Output Serial Monitor ×

Message (Enter to send message to 'ESP32 Wrover Module' on 'COM7')

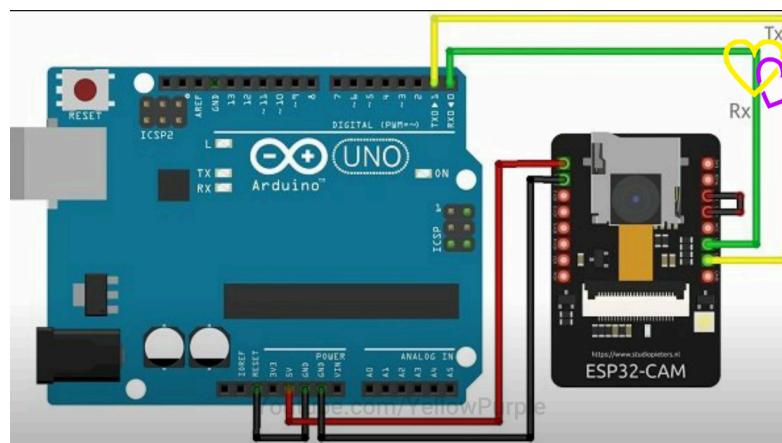
.....ets Jul 29 2019 12:21:46

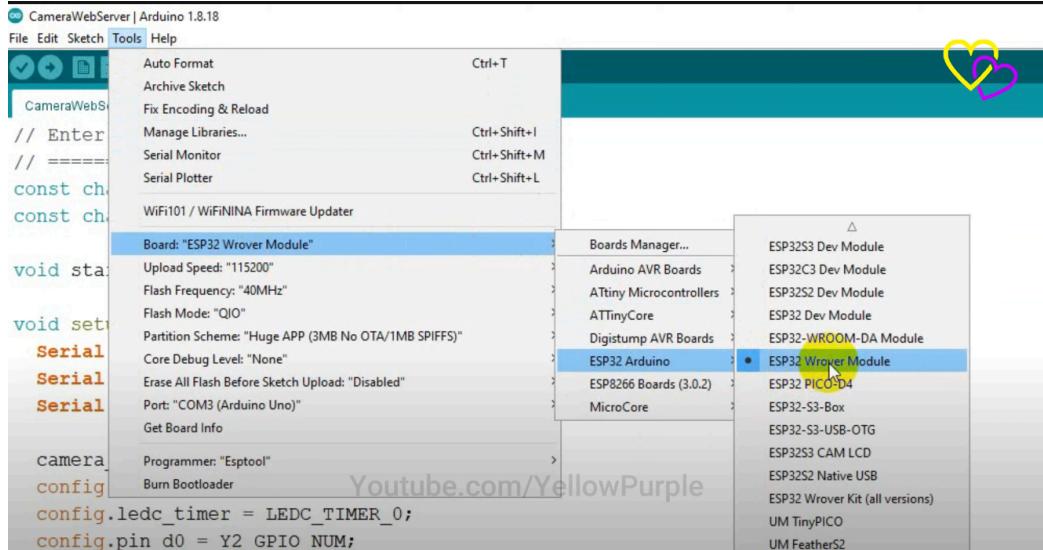
```

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config:ip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4832
load:0x40078000,len:16440
load:0x40080400,len:4
ho 8 tail 4 room 4
load:0x40080404,len:3504
entry 0x400805cc
E (396) esp_core_dump♦x♦+~♦ No core dump partition found!
E (396) esp_core_dump_flash: No core dump partition found!

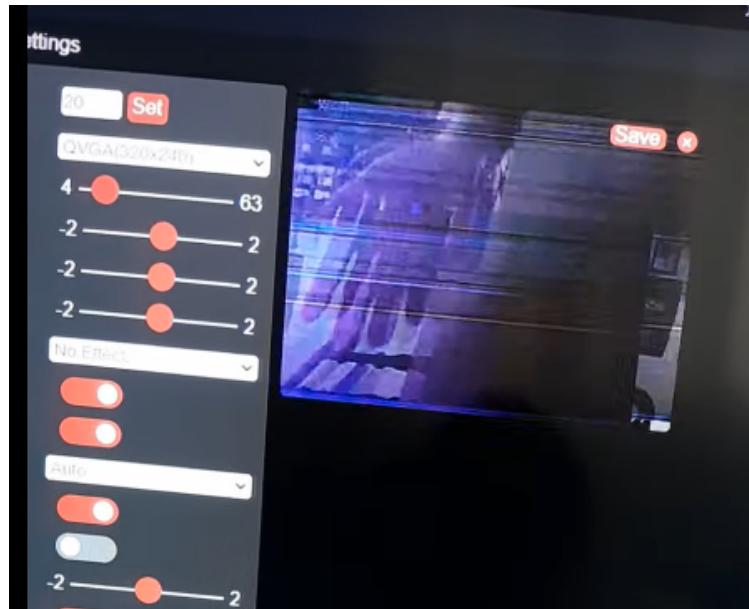
```

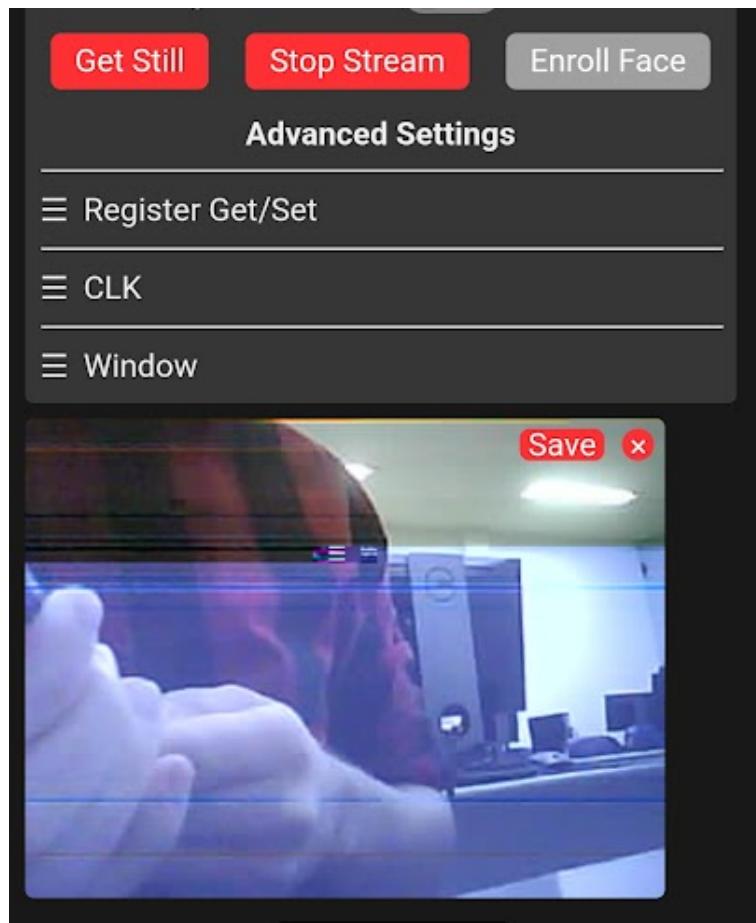
Além disso, a configuração da câmera demandava uma série de etapas minuciosas: "Após fazer as conexões de circuito adequadas entre o ESP32-CAM e o programador FTDI/Arduino, certifique-se de que as configurações adequadas estejam selecionadas. Por exemplo, selecione 'ESP32 Wrover Module' ou 'AI-Thinker ESP32-CAM' de acordo com o modelo da sua câmera ESP32 em Tools->Board. Além disso, escolha a porta correta para o Arduino ou o programador FTDI e certifique-se de que ela esteja conectada em Tools->Port. Quando o botão de upload for clicado e 'Connecting...' for apresentado na janela de depuração, pressione o botão de reset do ESP32-CAM. O esboço deve começar a ser carregado para o módulo da câmera".





Uma vez superada essa etapa, configurei um servidor local que transmitia imagens em tempo real da câmera pelo navegador. No entanto, deparei-me com outro problema: a imagem apresentava glitches. Naquele momento, não sabia se a falha era elétrica, de codificação, ou se a câmera estava com defeito.





Após consultar meu professor e utilizar um código que capturava uma única foto, concluí que o problema estava na câmera. Como não havia tempo hábil para solicitar a troca, tive que improvisar. A solução envolveu configurar múltiplas capturas, na esperança de que uma delas saísse sem erros visuais. Além disso, na captura agendada, estabeleci a tiragem de 10 fotos, aumentando as chances de obter uma com qualidade satisfatória.

Diante dessas dificuldades, optei por seguir com a captura de fotos unitárias, uma vez que a transmissão em tempo real se mostrou inviável devido à baixa performance da câmera. Em seguida, enfrentei novos desafios na comunicação entre o Arduino e o Python, principalmente no que se refere à transferência da imagem do ambiente físico para o lógico. Resolvi essa questão utilizando comunicação serial para enviar a imagem em base64, permitindo que o Python recebesse, decodificasse e processasse o arquivo diretamente.

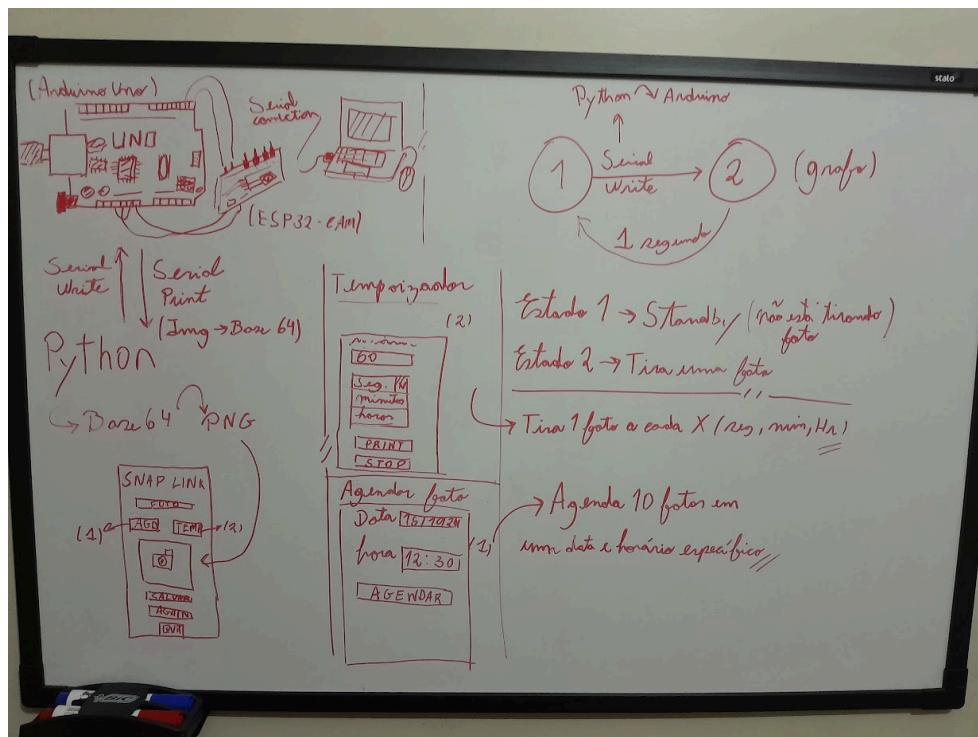
```
File Edit Selection View Go Run Terminal Help
sketch_oct14a
explorer ... main.py test_snaplink.py ...
main.py ...
89 draw.text(texto_posicoes) > base
90
91 def tirar_foto():
92     global ser
93     if ser and ser.is_open:
94         try:
95             ser.write(b't')
96             time.sleep(1)
97
98             base64_data = ""
99             while True:
100                 line = ser.readline().decode('utf-8').strip()
101                 if line == "#Início:":
102                     base64_data = ""
103                 elif line == "#Fim":
104                     break
105                 else:
106                     base64_data += line
107
108             print("Imagem em Base64 recebida.")
109             image_data = base64.b64decode(base64_data)
110
111             return image_data
112
113         except Exception as e:
114             messagebox.showerror("Erro", f"Erro ao capturar imagem: ({e})")
115
116     else:
117         messagebox.showerror("Erro", "Arduino não conectado.")
118
119     return None
120
121 def tirar_foto_unica():
122     image_data = tirar_foto()
123
124     if image_data:
125         mostrar_preview(image_data)
126
127     else:
128         messagebox.showerror("Erro", "Arduino não conectado.")
129
130     return image_data
131
132 def capturar_fotos_continuamente(intervalo):
133     global pasta_salvar
134     timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
135     file_name = f"foto_{timestamp}.jpg"
136     file_path = os.path.join(pasta_salvar, file_name)
137
138     adicionar_data_hora(image)
139     image.save(file_path)
140     print(f"Imagem salva automaticamente em: {file_path}")
141
142 def abrir_janela_temporizador():
143     global janela_temporizador
144     janela_temporizador = ctk.Toplevel(app)
145     janela_temporizador.geometry("400x300")
146     janela_temporizador.title("Captura com Temporizador")
147
148     temporizador_label = ctk.CTkLabel(janela_temporizador, text="Intervalo:")
149     temporizador_label.pack(pady=5)
150
151     global temporizador_intervalo_entry
152     temporizador_intervalo_entry = ctk.CTkEntry(janela_temporizador)
153     temporizador_intervalo_entry.pack(pady=5)
154
155     global unidade_temporizador
156     unidade_temporizador = ctk.CTkComboBox(janela_temporizador, values=[]
157     unidade_temporizador.set("Segundos")
158     unidade_temporizador.pack(pady=5)
159
160     btn_iniciar_captura = ctk.CTkButton(janela_temporizador, text="Iniciar")
161     btn_iniciar_captura.pack(padx=50, pady=10, fill='x')
162
163     btn_pausar_captura = ctk.CTkButton(janela_temporizador, text="Pausar")
164     btn_pausar_captura.pack(padx=50, pady=10, fill='x')
165
166     btn_voltar = ctk.CTkButton(janela_temporizador, text="Voltar")
167     btn_voltar.pack(padx=50, pady=10, fill='x')
168
169     janela_temporizador.mainloop()
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
```

The screenshot shows a Windows desktop environment with three main windows open:

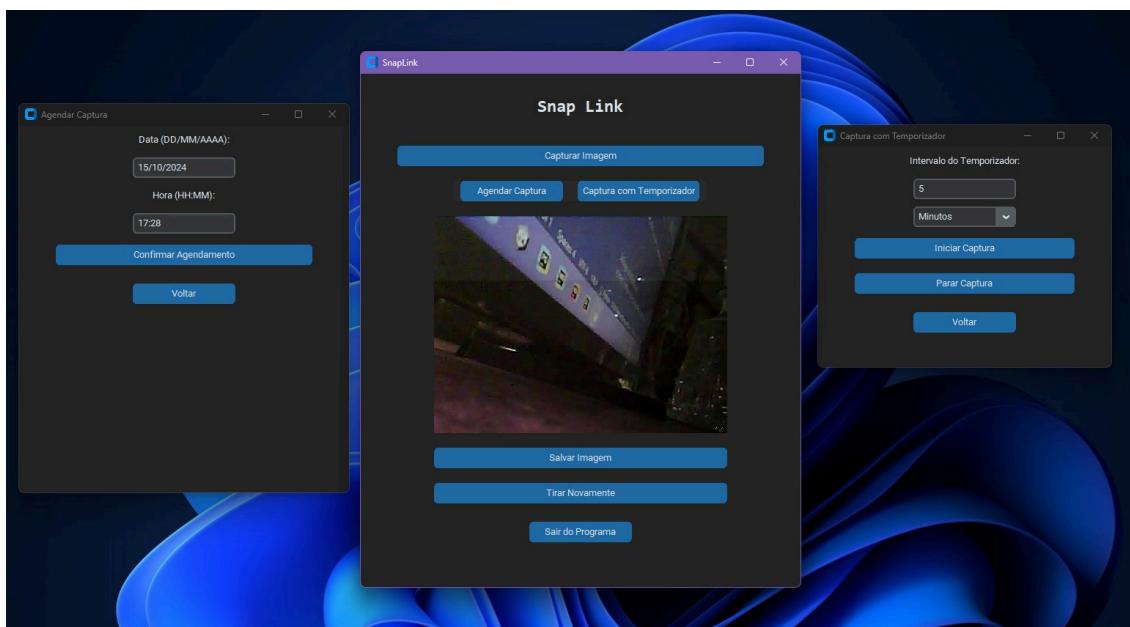
- Arduino IDE (Left Window):** The title bar says "sketch_0ct14a | Arduino IDE 2.3.2". It displays the code for the "ESP32 Wrover Module" sketch. The code includes functions for capturing images from the camera module and sending them via serial port. A button labeled "Conectar Arduino" is visible.
- Python Terminal (Top Right Window):** The title bar says "python_com.py - C:\Users\lemon\OneDrive\Desktop\python_com.py (3.10.1)". It shows the command-line interface for connecting to the Arduino over serial. A message box is displayed with the text "Conexão com Arduino estabelecida!" (Connection with Arduino established!).
- Terminal (Bottom Right Window):** The title bar says "CMD Shell 3.10.11". It shows the command-line interface for receiving data from the Arduino. The terminal output includes several "Writing at" messages followed by a "Hash of data verified." message. Below this, it says "Leaving..." and "Hard resetting via RTS pin...".

The taskbar at the bottom shows various pinned icons, including Microsoft Edge, File Explorer, and FileZilla. The system tray indicates the date as 15/08/2024 and the time as 11:26.

Ao perceber as limitações iniciais do projeto, decidi reformular minha abordagem. Resolvi, de última hora, criar uma interface gráfica em Python — uma área em que eu já possuía experiência e familiaridade. Realizei um brainstorm, rabiscando ideias em um quadro branco, para tornar o projeto mais criativo e funcional. Em apenas um dia, desenvolvi a GUI e implementei todas as funcionalidades planejadas, o que resultou em um protótipo funcional e bastante satisfatório. Posteriormente, apenas refinei alguns detalhes, corrigi pequenos erros, realizei testes unitários e implementei validações e tratamentos de exceção.



No fim das contas, essa experiência de trabalhar com tantas tecnologias foi extremamente enriquecedora. Aprendi muito ao longo do projeto e saí com um senso de realização e crescimento profissional.



3.2.4. REFLEXÃO APROFUNDADA

A experiência vivida durante o projeto reforçou a importância da comunicação eficaz e da colaboração com o cliente para o sucesso de um projeto. A capacidade de adaptar-se às necessidades em constante evolução do cliente foi fundamental para garantir a entrega de um produto final que atendesse às suas expectativas.

3.2.5. CONSIDERAÇÕES FINAIS

Olhando para o futuro, vejo oportunidades para aprimorar ainda mais o sistema desenvolvido e explorar novas soluções tecnológicas para ampliar seu impacto na comunidade. Pretendo continuar a aprimorar minhas habilidades técnicas e de comunicação, buscando enfrentar desafios ainda maiores no desenvolvimento de software e na interação com os clientes.

(Inserir imagens da apresentação)