EyeOnTree – Sistema IoT de Monitoramento de Florestas de Celulose Utilizando ESP32-CAM e Telegram

Jorge Lucas de Oliveira *

Kauê Balbino de Menezes **

Nicolas Henrique Izaias de Souza ***

Rafael Cecílio Gomes ****

Resumo:

O projeto em questão consiste em um sistema de monitoramento de florestas de celulose/silvicultura contra incêndios. Este sistema opera por meio do módulo câmera ESP32-CAM com inteligência artificial integrada, em conjunto com uma plataforma de prototipagem baseada em ESP32. A transmissão e recebimento de dados são realizados via conexão Wi-Fi entre os módulos ESP32-CAM, ESP32 e o aplicativo de mensagens Telegram.

Palavras-chave:

IoT; ESP32; ESP32-CAM; Monitoramento florestal; Eletrônica; I.A.

I. INTRODUÇÃO

"Em setembro de 2024, mais de 10 mil hectares de fazendas de eucalipto foram devastados por um incêndio em Água Clara (MS), a 193 km de Campo Grande. Após o

*Técnico em Eletrônica, Etec Philadelpho Gouvêa Netto – oliveirajorgelucas05@gmail.com
**Técnico em Eletrônica, Etec Philadelpho Gouvêa Netto – kbalbinodemenezes@gmail.com
***Técnico em Eletrônica, Etec Philadelpho Gouvêa Netto – nic.izaias2008@gmail.com
****Técnico em Eletrônica, Etec Philadelpho Gouvêa Netto – giccecilio@gmail.com

controle inicial, as chamas reacenderam com intensidade. O Corpo de Bombeiros, brigadistas locais e equipes da Força Nacional, além de bombeiros do Rio Grande do Sul, estiveram no local para reforçar a Operação Pantanal e conter o avanço do fogo." (PORTAL CELULOSE, 2024).

> *"Considerando o faturamento, segundo o boletim Casa Rural, a celulose foi o produto florestal mais exportado por Mato Grosso do Sul em 2024, entre janeiro e outubro, com participação de 98,89%. O segundo lugar ficou para papel com 1,01% e madeira com 0,10%. O total das exportações florestais chegou a US$ 2,145 bilhões, valor 79,7% maior que os US$ 1,193 bilhão exportados no mesmo período do ano anterior." (MAIS FLORESTA, 2025).*

Incêndios em áreas rurais têm se tornado cada vez mais frequentes. No ano de 2024, o Brasil enfrentou aproximadamente 280 mil focos de incêndio, de acordo com o INPE. Enquanto isso, o preço do m³ (metro cúbico) de celulose também vem aumentando. Logo não é impossível que uma produção de eucalipto, uma das espécies de árvore mais favoráveis para a produção de celulose, que leva aproximadamente de 6 a 8 anos para iniciar-se a coleta, sofra um incêndio no decorrer deste período e todo o material seja perdido.

Durante pesquisas, a equipe observou que boa parte dos produtores rurais e naturais não está familiarizada com as atuais tecnologias, como, por exemplo, Indústria 4.0 ou IoT (Internet das Coisas), talvez devido à falta de conexão, demanda ou orçamento. Portanto, não seria comovente ajudar somente a parcela apta a implementar essas inovações. Pensando em toda a população de produtores, criou-se o EyeOnTree, um sistema com a função de evitar que uma produção de celulose seja gravemente afetada por incêndios.

Objetivo geral

Realizar o monitoramento de florestas, áreas de cultivo e culturas em geral com imagens, para a prevenção de incêndios. Garantir a segurança e qualidade da produção. Melhorar a lucratividade do proprietário e aumentar a demanda do produto no mercado.

Objetivos específicos

Aprofundar o entendimento sobre o funcionamento da plataforma de prototipagem ESP32 Dev-Kit e introduzir estudos com ESP32-CAM.

Compreender as operações realizadas pelo módulo câmera e a forma como captura imagens.

Utilizar inteligência artificial para detectar fogo e incêndio nas imagens capturadas pela câmera.

Implementar o módulo LoRa (Long Range) para a conexão a longas distâncias.

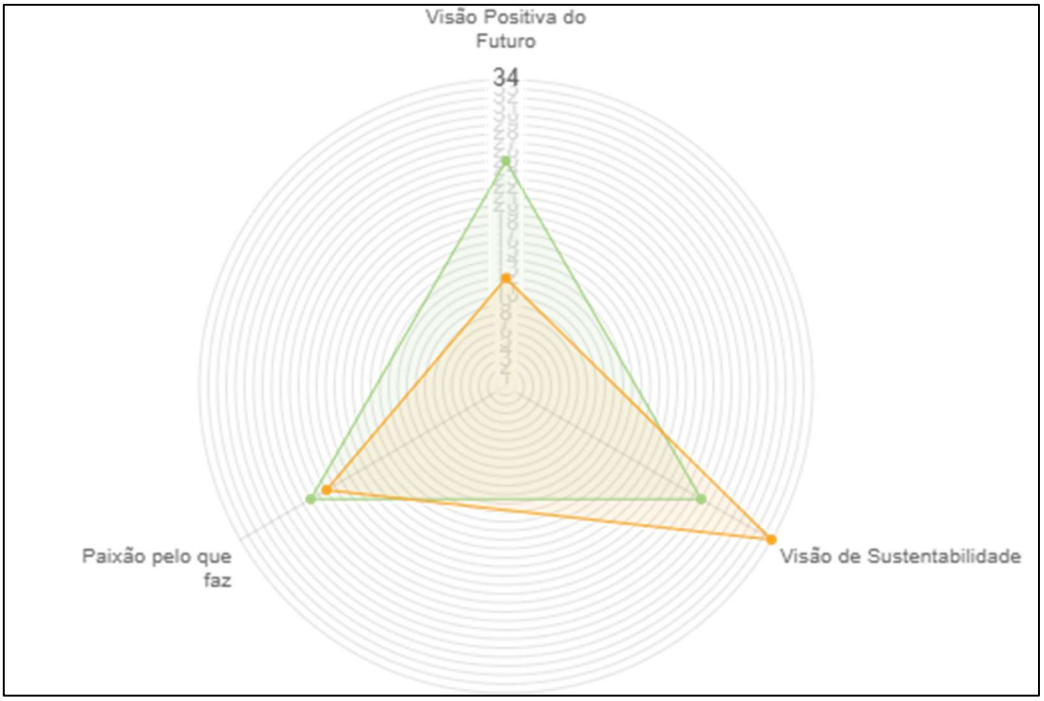
Estabelecer comunicação do EyeOnTree com o Telegram para as notificações de alerta para um smartphone.

II. DESENVOLVIMENTO

1. Desenvolvimento pessoal e profissional (eZAPe!)

Jorge Lucas de Oliveira:

Figura 1 - Gráfico de autoavaliação de Jorge Lucas de Oliveira

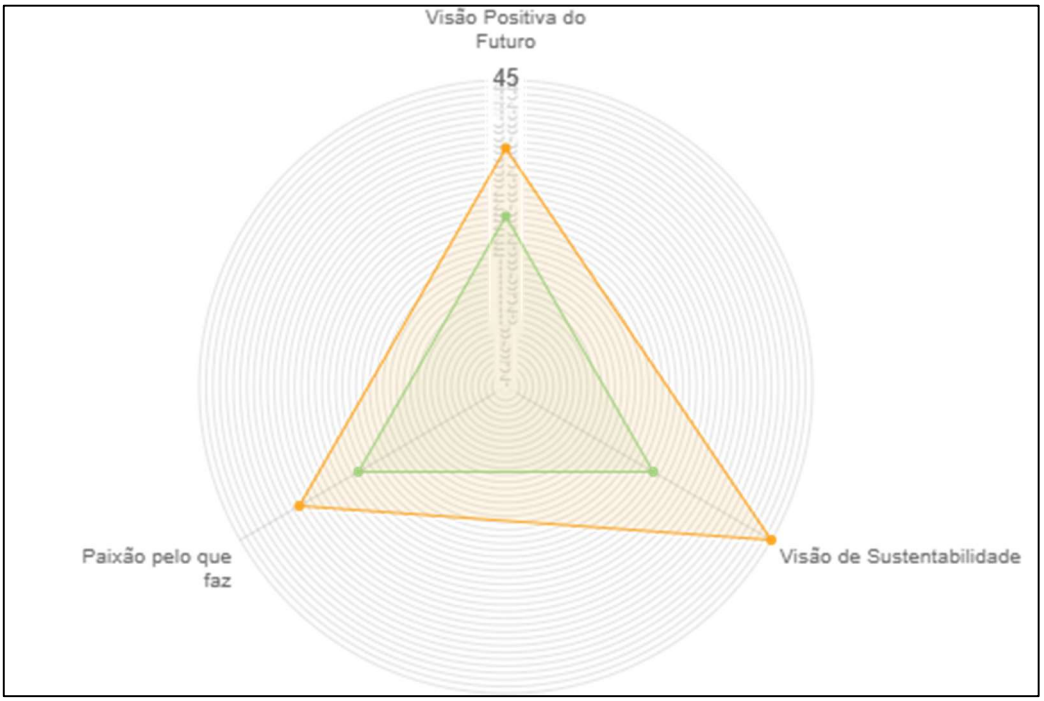


Crédito: De autoria própria utilizando eZAPe! / IAMAR 2025

“Participar da plataforma “eZAPe!” foi uma das experiências que mais abriram a minha cabeça sobre o empreendedorismo. Aprendi a enxergar oportunidades onde antes eu via só problemas, a entender melhor como funciona a construção de um negócio e, principalmente, a pensar de forma mais estratégica. Hoje me sinto mais preparado para tomar decisões, organizar minhas ideias e transformar projetos em realidade. A “eZAPe!” realmente fez diferença no meu desenvolvimento.”

Kauê Balbino de Menezes:

Figura 2 - Gráfico de autoavaliação de Kauê Balbino de Menezes

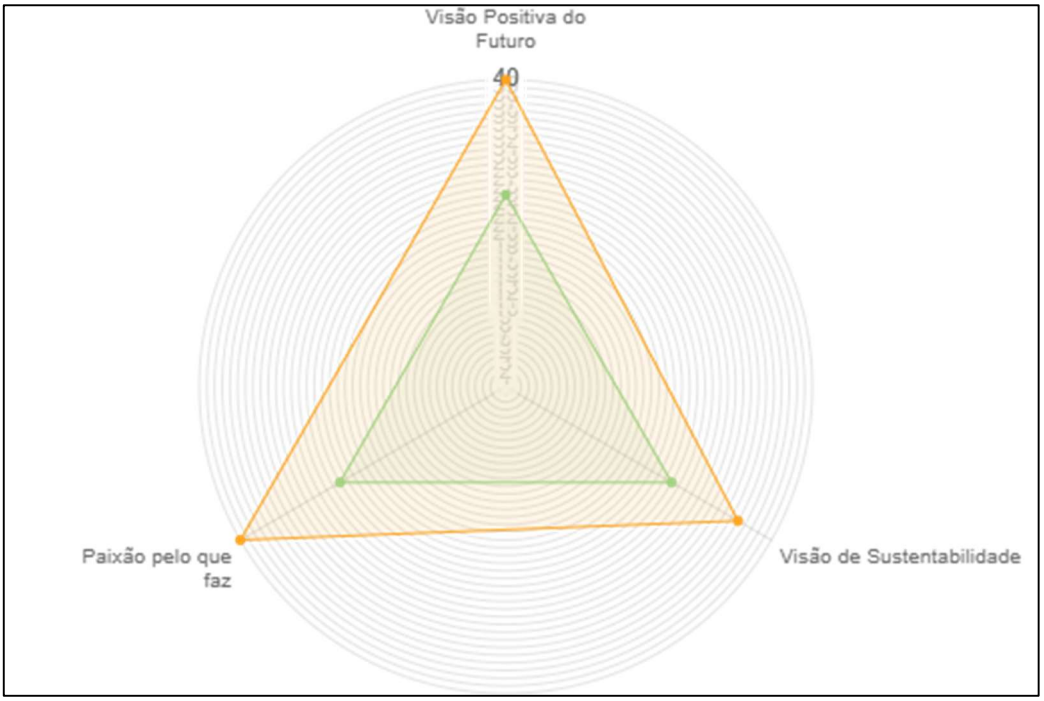


Crédito: De autoria própria utilizando eZAPe! / IAMAR 2025

“Neste último ano escolar, aprendi a reconhecer as oportunidades que surgiam e a aproveitá-las. Muito disso veio graças ao “eZAPe!”, que me proporcionou uma nova visão sobre empreendedorismo. Além disso, precisei aprender a lidar com pessoas em um ambiente profissional, especialmente por meio das relações que desenvolvi com meus colegas durante a elaboração do nosso TCC. Também tive de aprender a organizar melhor meu tempo para equilibrar as diferentes áreas da minha vida, como o projeto do TCC e os estudos para o ENEM.”

Nicolas Henrique Izaias de Souza:

Figura 3 - Gráfico de autoavaliação de Nicolas Henrique Izaias de Souza



Crédito: De autoria própria utilizando eZAPe! / IAMAR 2025

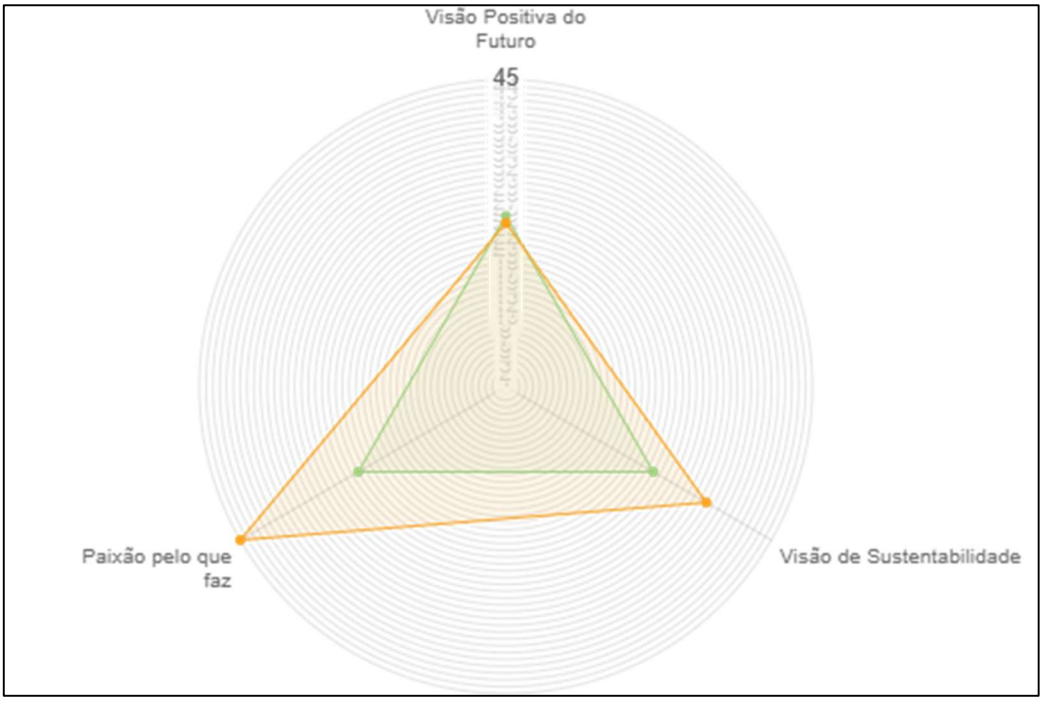
“Esse último ano do Ensino Médio, em relação aos dois primeiros, foi cheio de mudanças. Passei a enxergar mais facilmente as oportunidades, bem como ter bom proveito do curso técnico e ser proativo, ao menos mais, em relação ao meu eu anterior. Consegui, com ajuda de plataformas como o “eZAPe!” e de conversas e convívio com professores e família, entender o que é preciso para se desenvolver. Vi o quão é importante ter pessoas como inspiração, motivação. Ter pessoas que acreditem em seu potencial e te apoiem quando for preciso. Procurando sempre aprender algo novo ou melhorar, para não me estagnar.

Ainda com utilização da plataforma de empreendedorismo, além da prática de virtudes e morais, desenvolvi paixão pelo ato de desejar o bem de outrem, sempre agradecendo a Deus e a tudo que Ele nos proporciona.

Profissionalmente, compreendi como traçar sonhos e metas é essencial. Considerando isso, comecei a enfatizar meus objetivos e me informar do que é preciso para alcançá-los, dispondo-me a fazê-lo.”

Rafael Cecílio Gomes:

Figura 4 - Gráfico de autoavaliação de Rafael Cecílio Gomes



Crédito: De autoria própria utilizando eZAPe! / IAMAR 2025

“Para mim a participação na plataforma “eZAPe!” contribuiu significativamente para ampliar a visão sobre criação e desenvolvimento de projetos. A experiência ajudou a compreender melhor como transformar desafios em oportunidades e observar o processo de planejamento com mais clareza. Além disso, ofereceu uma base mais sólida para organizar ideias e direcionar ações de maneira objetiva. No geral, o uso da plataforma representou um avanço importante na forma de pensar e estruturar projetos."

2. Desenvolvimento empreendedor

O EyeOnTree visa proporcionar conforto e segurança para produtores rurais, por meio de um sistema de monitoramento de baixo custo para as culturas. Esse sistema tem como objetivo evitar perdas graves causadas por incêndios em florestas de silvicultura e plantações, contribuindo para o aumento da produtividade e proteção ao produtor.

A empresa busca estabelecer uma relação de fidelidade e transparência com seus clientes, oferecendo solução das questões relacionadas a incêndios em áreas rurais. Isso será alcançado por meio do produto, bem como a comunicação ativa e direta do EyeOnTree com o público.

Figura 5 - Modelo de negócios / Canvas

Crédito: De autoria própria utilizando eZAPe! / IAMAR, 2025

O grupo de clientes é formado por produtores rurais, como agricultores, agrônomos, floricultores, mas principalmente produtores de eucalipto para celulose e seringueira para látex, além de empresas fabricantes de derivados destes produtos, como, por exemplo, papel e borracha. Para atingir o público, realizaremos divulgações on-line em redes sociais, bem como a participação em feiras e exposições técnicas e o contato com grandes empreendimentos, para melhor conhecimento e compartilhamento, incentivando os clientes sobre a importância da segurança e a experiência de utilizar novos métodos de prevenção contra incêndios.

Os parceiros-chave da empresa incluem principalmente fábricas e empresas, sendo possível até empresas concorrentes, que possam compartilhar dados e favorecer progresso mútuo.

A atividade principal da empresa consiste na avaliação do funcionamento do sistema. Trata-se de um suporte abrangente para vistoriar o bom funcionamento do produto. Além disso, a transparência com os clientes em relação a atualizações, erros e informações relacionadas ao dispositivo é essencial. A empresa conta com assistência ao cliente, uma equipe de especialistas e técnicos altamente capacitados para o desenvolvimento e manutenção dos produtos, bem como o ato de estabelecer uma relação concreta com os clientes e fornecedores.

A respeito dos recursos essenciais, necessita-se de microcontroladores, sensores de temperatura, servo-motores, jumpers, materiais para construção da torre e internet para a programação do produto.

O conhecimento na área da eletrônica será fundamental para desenvolver a comunicação do ESP32 com o aplicativo (Telegram), como na área de programação, lidando com softwares (IDE Arduino, Virtual Studio Code e entre outros) e linguagens de

programação. Estabelecer parcerias, buscar patrocínios e lucros provenientes dos clientes será crucial.

O aspecto financeiro da empresa é sustentado pelas vendas dos produtos e pelas mensalidades dos serviços oferecidos.

Com o auxílio do DreamShaper, foi elaborada uma tabela abrangente de despesas.

É importante ressaltar que os valores são fictícios e foram estabelecidos com base nas tendências do mercado atual.

A tabela subsequente apresenta de forma abrangente todos os gastos, desde despesas até tributos. O resultado líquido é positivo, evidenciando uma tendência ascendente ao longo dos anos.

Figura 6 - Tabela de despesas

	TEMPO		
	1 ano	2 anos	3 anos
Receitas (R)	23.600,00 R$	47.000,00 R$	130.000,00 R$
Custos Variáveis (CV)	5.000,00 R$	5.000,00 R$	5.000,00 R$
Custos Fixos (CF)	1.100,00 R$	1.100,00 R$	1.100,00 R$
Custos Totais (CT = CV + CF)	6.100,00 R$	6.100,00 R$	6.100,00 R$
Resultado Operacional (RO = R - CT)	17.500,00 R$	40.900,00 R$	123.900,00 R$
Depreciação e Amortização (DA)	8.750,00 R$	11.250,00 R$	12.250,00 R$
Resultado Antes de Impostos (RAI = RO - DA)	8.750,00 R$	29.650,00 R$	111.650,00 R$
Impostos (I)	2.581,25 R$	8.746,75 R$	32.936,75 R$
Resultado Líquido (RL = RAI - I)	6.168,75 R$	20.903,25 R$	78.713,25 R$

Crédito: De autoria própria utilizando eZAPe! / IAMAR, 2025

Os “Custos Variáveis” correspondem à construção do produto e a manutenção.

Os “Custos Fixos” englobam os salários dos funcionários e o material para a construção do protótipo.

É notável o crescimento exponencial do lucro, alcançado principalmente pela venda do produto e com manutenções periódicas que a nossa equipe vai oferecer e realizar.

Por fim, uma identidade visual para o EyeOnTree é estabelecida, incorporando as cores: verde-escuro (#06653A), verde-bandeira (#097550) e branco (#FFFFFF), em conjunto com o slogan “protegendo florestas com inteligência”. Essa abordagem confere à empresa uma imagem de proteção e tecnologia para seus clientes, reforçando a finalidade do produto: a segurança das florestas.

Figura 7 - Logo do EyeOnTree

Crédito: De autoria própria utilizando Capcut

Figura 8 - Logo do EyeOnTree

Crédito: De autoria própria utilizando Capcut

3. Metodologia do projeto

O projeto teve seu tema estabelecido após uma série de pesquisas, a fim de identificar áreas do mercado atual em que houvesse problemas. Escolheu-se silvicultura, vistos recentes acontecimentos de incêndios em florestas.

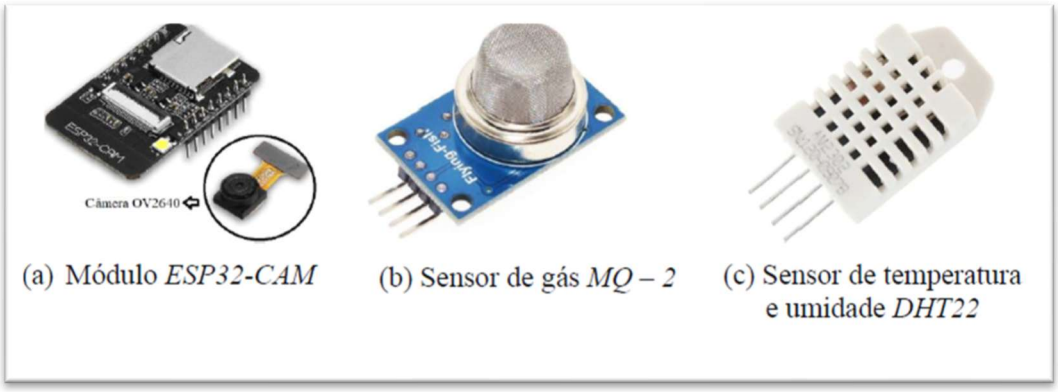
Concretizado o conceito, foram encontrados trabalhos anteriores com ideias parecidas de estudantes de universidades e faculdades do estado e exteriores.

Figura 9 - ESP32-CAM salva fotos em Firebase

Crédito: https://randomnerdtutorials.com/esp32-cam-save-picture-firebase-storage/

A imagem acima trata-se da capa de um site/blog do Random Nerd Tutorials. O tema cuja imagem foi retratada é a utilização do microcontrolador ESP32-CAM para a captura de imagens e salvamento destas em uma nuvem, no caso utilizado o sistema Firebase.

Figura 10 - Proposta de sistema de monitoramento de incêndios ambientais utilizando o ESP32-CAM



Crédito: https://sol.sbc.org.br/index.php/eri-mt/article/view/18229/18063

A figura 10 faz referência a um artigo científico de alunos do Instituto Federal de Mato Grosso (IFMT). Foi elaborada a proposta de um sistema de monitoramento de incêndios ambientais, utilizando o ESP32-CAM para o monitoramento. Para isso, realizaram-se pesquisas de campo e teóricas em relação às mudanças climáticas e ao aquecimento global. O EyeOnTree visa colocar em prática esses conceitos, concretizando um sistema funcional que resolva problemas referentes a perdas e prejuízos financeiros, considerando causas naturais como segundo plano.

Figura 11 - DIY AI Camera with Google Vision & ESP32 CAM Module

Crédito: https://how2electronics.com/diy-ai-camera-with-google-vision-esp32-cam-module/

A figura 11 faz referência a um blog, que realiza a produção de um projeto em que o ESP32-CAM realiza captura de imagens e as informações destas são representadas em

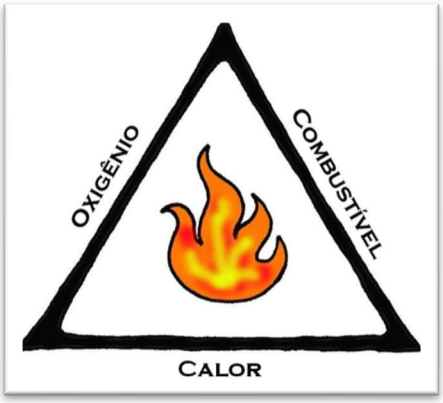
um display TFT. Tal conhecimento foi essencial para a elaboração do funcionamento do projeto, desconsiderando somente a utilização da tela.

4. Fundamentação teórica

As plantações de celulose, compostas principalmente por espécies como eucalipto e pinus, representam uma atividade econômica de grande relevância para o Brasil. Contudo, essas áreas apresentam alta suscetibilidade a incêndios florestais, fenômenos capazes de gerar sérios prejuízos econômicos, ambientais e operacionais. De acordo com o Sistema Nacional de Prevenção e Combate aos Incêndios Florestais (PREVFOGO/IBAMA), esses eventos caracterizam-se pela rápida e desordenada propagação do fogo, comprometendo vastas áreas e trazendo consequências significativas tanto para o setor florestal quanto para o meio ambiente.

O incêndio florestal ocorre a partir da combustão descontrolada da vegetação, exigindo a presença de três elementos fundamentais, conhecidos como o “triângulo do fogo”: combustível, oxigênio e calor.

Figura 12 - Ilustração do triângulo do fogo



Crédito: https://bushcraftberg.blogspot.com/2014/03/triangulo-do-fogo.html

O combustível corresponde ao material vegetal inflamável, como folhas secas, galhos, cascas, madeira e capim. Nas plantações de celulose, esse fator é agravado pelo acúmulo de serapilheira no solo e pela presença de óleos essenciais nas folhas de eucalipto, substâncias voláteis que intensificam a inflamabilidade. O oxigênio, presente naturalmente na atmosfera em cerca de 21%, garante a manutenção da combustão, sendo suficiente para alimentar o fogo mesmo em condições de calmaria. Já o calor, considerado a fonte de ignição, pode ter origem em ações humanas — como bitucas de cigarro, queima de lixo, faíscas de máquinas agrícolas, fogueiras, queimadas mal

controladas e até atos de vandalismo — ou em causas naturais, como raios ou combustão espontânea em áreas secas e com grande acúmulo de material vegetal.

O desenvolvimento de um incêndio florestal segue um processo que pode ser dividido em quatro etapas principais. A primeira é a ignição, momento em que a combinação dos elementos do triângulo do fogo resulta no início das chamas. Em seguida ocorre a propagação, caracterizada pelo rápido espalhamento do fogo, favorecido por ventos fortes, baixa umidade relativa do ar e abundância de vegetação seca. Se não controlado, o incêndio entra na fase de intensificação, aumentando sua velocidade, temperatura e área atingida. Por fim, chega-se à extinção, que pode acontecer naturalmente, com a ocorrência de chuvas ou pela redução do combustível, ou de forma artificial, com o uso de aceiros, brigadas de incêndio, abafadores, aeronaves e recursos hídricos.

Diversos fatores aumentam o risco de incêndios em plantações de celulose. Entre os mais relevantes estão o clima seco e quente, a baixa umidade relativa do ar, a presença de ventos fortes, as estiagens prolongadas e o manejo inadequado de resíduos florestais, como galhos, folhas e cascas deixados no solo. Esses elementos, quando combinados, criam um ambiente altamente propício à ignição e propagação do fogo.

Os prejuízos resultantes de incêndios florestais em áreas de produção de celulose podem ser divididos em econômicos, ambientais e operacionais.

- **Prejuízos Econômicos**

Do ponto de vista econômico, os impactos são severos. A perda de produção é significativa, já que árvores que levaram anos para atingir o ponto de corte são destruídas, comprometendo a colheita e a cadeia produtiva. Além disso, há altos custos com combate e prevenção, como a manutenção de brigadas especializadas, aquisição de equipamentos, abertura de aceiros, instalação de torres de observação, sistemas de monitoramento e uso de drones. Também há risco de danos à infraestrutura, como viveiros, galpões, maquinários e estradas internas, bem como possibilidade de multas ambientais caso se comprove negligência na prevenção. Em casos extremos, um incêndio que atinja 100 hectares de eucalipto pronto para corte pode gerar prejuízos milionários e atrasar significativamente a produção industrial.

- **Prejuízos Ambientais**

Os prejuízos ambientais também são expressivos. O fogo provoca a emissão de gases do efeito estufa, como CO_2, CO e NOx, contribuindo para o aquecimento global. Além disso, há degradação do solo, perda de nutrientes, compactação e aumento da suscetibilidade à erosão. A fauna local é fortemente impactada, com a morte de animais e consequente desequilíbrio ecológico. A poluição do ar e da água também é intensificada pela presença de fuligem, cinzas e compostos tóxicos.

- **Prejuízos Operacionais**

No âmbito operacional, os incêndios dificultam a gestão das plantações. As atividades de colheita, transporte e replantio podem ser interrompidas, colocando em risco a integridade física de trabalhadores e brigadistas. O ciclo de produção precisa ser replanejado, exigindo novos investimentos e atrasando metas previamente estabelecidas.

- **Síntese dos Impactos**

Em síntese, um incêndio florestal pode comprometer todo o ciclo produtivo do eucalipto, que dura de seis a sete anos entre o plantio e o corte. Dessa forma, em poucas horas, é possível perder o investimento de anos de trabalho, além de milhões de reais.

Diante dos riscos e impactos dos incêndios florestais, as tecnologias de monitoramento têm se mostrado fundamentais para a prevenção e o combate eficiente. Uma dessas soluções é o EyeOnTree, sistema inteligente de monitoramento que atua na detecção automática de indícios de fogo em tempo real, permitindo resposta rápida por parte das equipes de campo.

Os benefícios dessa tecnologia incluem a redução do tempo entre a detecção e a ação de combate, a diminuição das perdas financeiras por evitar a destruição de grandes áreas, a maior segurança de trabalhadores e da infraestrutura e a contribuição para a sustentabilidade por meio do manejo responsável e da mitigação dos impactos ambientais. O sistema pode ser integrado a torres de monitoramento, câmeras térmicas, sensores atmosféricos e inteligência artificial, garantindo uma vigilância eficiente e menos dependente da observação humana contínua.

5. Protótipo, montagem e testes

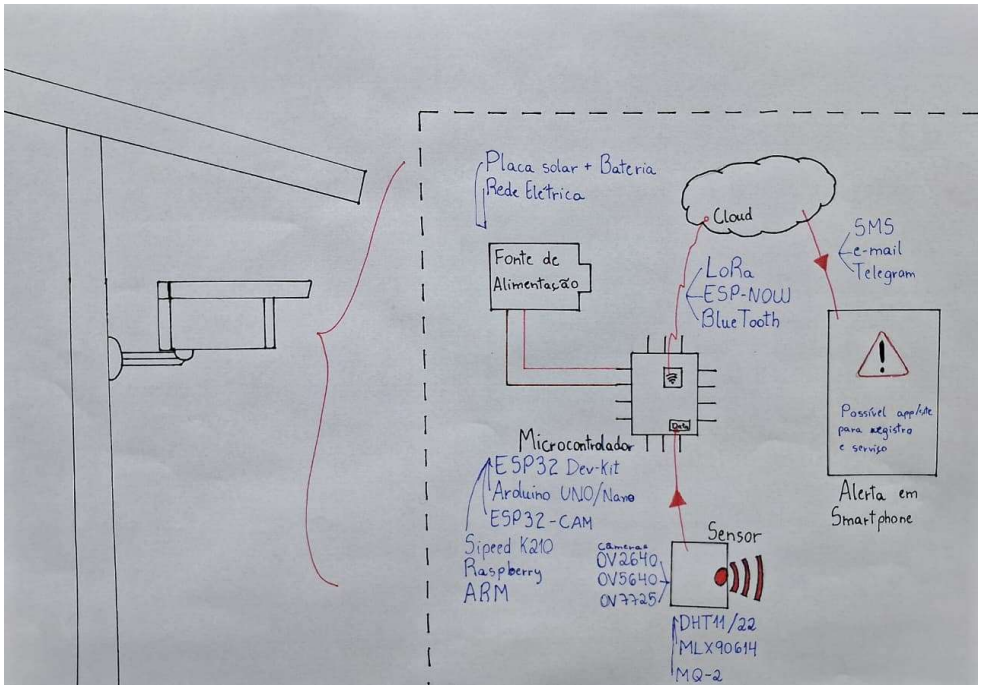
A estrutura do EyeOnTree consiste em uma câmera com inteligência artificial integrada, instalada a uma torre. Se essa detectar uma anomalia, como fogo ou incêndio, o sistema enviará uma notificação para um smartphone previamente configurado, cabendo ao seu usuário contatar autoridades para o combate do incêndio.

Figura 13 – Visão geral EyeOnTree

Crédito: Autoria própria – imagem digitalizada com I.A. 2025-17/06

O diagrama de blocos de EyeOnTree tem a finalidade de demonstrar as possibilidades de componentes a serem utilizados no projeto, bem como ilustrar a forma de comunicação entre os componentes. Os âmbitos a serem analisados são: microcontrolador, fonte de alimentação, sensor e comunicação.

Figura 14 – Diagrama de blocos EyeOnTree



Crédito: Autoria própria – desenho à mão, 2025-19/08

Microcontrolador:

- **Arduino Uno**

O Arduino Uno é um dos microcontroladores mais famosos e amplamente utilizados no mundo da prototipagem eletrônica. Sua simplicidade e o grande suporte comunitário são suas principais vantagens. Equipado com o ATmega328P, um microprocessador de 8 bits com 16 MHz de frequência, o Arduino Uno não possui o poder de processamento de outros microcontroladores, como o ESP32. Sua memória é bastante limitada, com apenas 2 KB de RAM e 32 KB de Flash.

O Uno possui 14 pinos digitais de entrada e saída e 6 pinos analógicos, além de oferecer suporte para comunicação UART, I2C e SPI. No entanto, ele não possui conectividade sem fio integrada. Para isso, seria necessário adicionar módulos extras, como o ESP8266 para Wi-Fi ou Bluetooth, o que aumenta o custo e a complexidade do projeto.

Por essas limitações, o Arduino Uno é mais indicado para projetos simples ou para quem está começando na eletrônica. Embora seja possível realizar muitos projetos interessantes com o Uno, sua falta de conectividade nativa e as limitações de memória e processamento fazem dele uma escolha menos vantajosa para projetos mais avançados que exigem conectividade sem fio ou maior poder de processamento.

- **Raspberry Pi**

O Raspberry Pi, ao contrário dos microcontroladores mencionados, é um minicomputador completo. Equipado com um processador ARM Cortex-A72 (quad-core de 1.5 GHz no modelo Raspberry Pi 4), ele oferece uma performance muito superior à de microcontroladores como o Arduino e o ESP32, especialmente em tarefas que envolvem processamento intensivo, como computação gráfica e execução de sistemas operacionais completos (como o Linux).

O Raspberry Pi também possui uma memória considerável, com opções que variam de 1 GB a 8 GB de RAM. Ele inclui portas USB, saída HDMI, Ethernet e Wi-Fi (a partir do modelo Raspberry Pi 3), o que o torna uma excelente escolha para projetos que precisam de conectividade de rede, interfaces gráficas ou até mesmo desenvolvimento de sistemas completos.

Entretanto, o Raspberry Pi não é ideal para todos os tipos de projetos. Seu alto consumo de energia e o fato de ser um computador completo fazem dele uma opção menos eficiente para projetos simples ou para aqueles que necessitam de baixo consumo de

energia. Ele é mais adequado para tarefas que envolvem processamento gráfico ou funcionamento de sistemas operacionais, como em servidores ou aplicações de computação mais complexas.

- **ARM**

A família ARM, especificamente os microcontroladores baseados na arquitetura Cortex-M (como os modelos STM32), é muito utilizada em aplicações industriais e em projetos que exigem alta performance e baixo consumo de energia. Com processadores de 32 bits e uma variedade de opções de clock (geralmente variando de 48 MHz a 180 MHz), os microcontroladores ARM são ótimos para tarefas que exigem precisão e desempenho em tempo real, como controle de sistemas embarcados.

Além disso, o STM32 possui uma grande flexibilidade em relação às entradas e saídas (GPIOs), com suporte a PWM, ADC, DAC, I2C, SPI e UART, entre outros. A memória varia de acordo com o modelo, mas geralmente está na faixa de 64 KB a 1 MB de Flash e 256 KB de RAM.

O principal desafio ao trabalhar com microcontroladores ARM é a curva de aprendizado. Eles exigem um conhecimento mais profundo sobre programação em baixo nível e sobre o funcionamento da arquitetura do microcontrolador. No entanto, para projetos industriais ou em tempo real, o ARM é uma excelente escolha pela sua eficiência energética e performance robusta.

- **Sipeed K210**

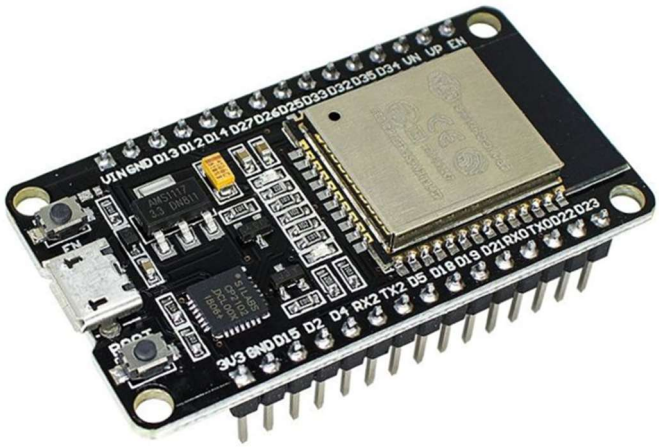
O Sipeed K210 é um microcontrolador menos conhecido, mas que vem ganhando destaque em projetos de inteligência artificial (IA), especialmente devido ao seu suporte nativo para rede neural artificial (AI). Equipado com um processador dual-core 64-bit RISC-V, ele oferece uma capacidade de processamento bastante alta, com clock de 1.0 GHz, o que o torna ideal para tarefas como reconhecimento de imagens, classificação de objetos e processamento de sinais.

O K210 possui 8 MB de RAM e 16 MB de memória Flash e oferece uma gama de entradas e saídas (GPIOs), com suporte a I2C, SPI, UART, entre outros. No entanto, ele não possui conectividade sem fio integrada, o que pode exigir o uso de módulos adicionais para comunicação.

Este microcontrolador é particularmente interessante para projetos de visão computacional e inteligência artificial embarcada. Embora seja uma ótima escolha para projetos com IA, o Sipeed K210 pode não ser a melhor opção para projetos que exijam conectividade sem fio ou baixo consumo de energia, já que ele não é tão otimizado para essas necessidades.

- **Espressif ESP32:**

Figura 15 - Espressif ESP32



Crédito: https://www.autocorerobotica.com.br/placa-de-desenvolvimento-wifi-bluetooth-esp32

Analisando tais características, escolheu-se o ESP32, um dispositivo de IoT (Internet das Coisas). Composto por um microprocessador dual-core de baixa potência Tensilica Xtensa de 32 bits LX6, que oferece suporte embutido para rede Wi-Fi, Bluetooth v4.2 e memória flash integrada, sua arquitetura possibilita a programação independente do ESP32, dispensando a necessidade de placas microcontroladoras adicionais como o Arduino. Suas principais características incluem baixo consumo de energia, alto desempenho, amplificador de baixo ruído, robustez, versatilidade e confiabilidade.

  - **Design Robusto:**

O ESP32 demonstra confiabilidade em ambientes industriais, operando em temperaturas que variam de -40 °C a +125 °C. Equipado com circuitos de calibração avançados, o ESP32 é capaz de dinamicamente corrigir imperfeições no circuito externo e adaptar-se a mudanças nas condições ambientais.

  - **Consumo de Energia Ultrabaixo:**

Desenvolvido para dispositivos móveis, eletrônicos vestíveis e aplicações de IoT, o ESP32 atinge níveis de consumo de energia notavelmente baixos por meio da utilização de uma combinação de software proprietário. Além disso, o ESP32 incorpora

características avançadas, como o refinamento do clock gating, diversos modos de energia e uma escala dinâmica de potência.

- **Alta Integração:**

O ESP32 apresenta uma integração avançada, incluindo interruptores de antena embutidos, balun RF, amplificador de potência, amplificador de recepção de baixo ruído, filtros e módulos de gerenciamento de energia. Esta integração aprimorada apresenta funcionalidades e versatilidade valiosas às aplicações, com requisitos mínimos de PCB.

- **Chip Híbrido Wi-Fi e Bluetooth:**

O ESP32 pode operar como um sistema autônomo completo ou como um dispositivo escravo de um MCU host, minimizando a sobrecarga na pilha de comunicação do processador principal do aplicativo. O ESP32 é capaz de interagir com outros sistemas para oferecer funcionalidades Wi-Fi e Bluetooth com auxílio das interfaces SPI/SDIO ou I2C/UART.

Sensores:

- **DHT11**

O DHT11 é um sensor amplamente utilizado para medir a temperatura e a umidade do ar, principalmente em projetos simples de eletrônica e automação. Sua popularidade se deve, em grande parte, ao baixo custo e à facilidade de integração com microcontroladores como o Arduino e o Raspberry Pi. No entanto, embora seja uma escolha comum, o DHT11 apresenta limitações que devem ser consideradas dependendo da aplicação.

Em termos de faixa de medição, o DHT11 é capaz de medir temperaturas entre 0°C e 50°C e umidade relativa entre 20% e 80%. Embora isso seja suficiente para muitos cenários cotidianos, ele não é adequado para ambientes que exigem medições em condições extremas, como climas muito frios ou quentes. Além disso, sua precisão deixa a desejar: o sensor tem uma margem de erro de ±2°C para a temperatura e ±5% para a umidade, o que significa que, em situações onde a precisão é crucial, ele pode não ser a melhor opção.

Outro ponto que pode ser um desafio para o DHT11 é sua velocidade de leitura. Enquanto sensores mais avançados, como o DHT22, conseguem realizar medições com maior rapidez, o DHT11 pode levar mais tempo para fornecer uma leitura confiável. Em

sistemas que exigem medições rápidas e contínuas, o DHT11 pode acabar não atendendo às expectativas.

Por outro lado, o DHT11 apresenta algumas vantagens importantes. O custo é um dos maiores atrativos. Se a prioridade for um sensor de temperatura e umidade acessível, o DHT11 é, sem dúvida, uma escolha acertada. Ele também é muito fácil de usar, com um protocolo de comunicação simples que exige apenas um fio para transmissão de dados. Essa simplicidade o torna ideal para iniciantes e para protótipos de baixo custo.

Além disso, o DHT11 é bastante compacto e tem um baixo consumo de energia, características que o tornam uma boa opção para projetos alimentados por bateria, como dispositivos portáteis ou sistemas de monitoramento de baixo consumo. Ele também é fácil de encontrar em kits educacionais, sendo uma excelente ferramenta para quem está começando no mundo da eletrônica.

Em termos de aplicações, o DHT11 é frequentemente utilizado em projetos simples de automação residencial, monitoramento ambiental em pequena escala, sistemas de cultivo como hidroponia e até em estações meteorológicas caseiras. Nessas situações, em que a precisão não é o fator mais crítico, o DHT11 é mais que suficiente para fornecer informações sobre as condições ambientais.

Porém, se a necessidade for de maior precisão ou uma faixa de medição mais ampla, o DHT11 pode não ser o ideal. Nesse caso, sensores como o DHT22 ou o BME280 podem ser mais adequados. O DHT22, por exemplo, oferece uma faixa maior de temperatura e umidade, além de maior precisão, embora a um custo mais elevado. O BME280, além de medir temperatura e umidade, também é capaz de medir pressão atmosférica, tornando-se mais adequado para aplicações em ambientes nos quais as condições meteorológicas precisam ser monitoradas de maneira mais detalhada.

- **MLX90614**

O MLX90614 é um sensor de temperatura infravermelho fabricado pela Melexis, e é bastante utilizado em diversas áreas que exigem medições precisas de temperatura sem contato físico. Seu principal diferencial é a capacidade de medir a temperatura de objetos ou superfícies a uma certa distância, sem a necessidade de tocá-los. Para isso, o sensor detecta a radiação infravermelha emitida pelos corpos, que é diretamente proporcional à sua temperatura. Isso o torna ideal para aplicações cujo contato físico com o objeto não

é viável, como em processos industriais ou na medição da temperatura do corpo humano.

O funcionamento do MLX90614 se baseia no princípio da radiação infravermelha. Todo corpo emite radiação IR, e a quantidade de radiação emitida é um indicativo de sua temperatura. O sensor possui uma célula fotodetectora capaz de captar essa radiação, e, então, a partir dessa leitura, calcula a temperatura do objeto. Como é um sensor sem contato, ele oferece a vantagem de medir a temperatura de maneira rápida, segura e sem risco de contaminação, já que não há necessidade de interação direta com o objeto.

Entre as suas características mais notáveis, destaca-se a faixa de medição de -70°C a 380°C, com precisão que pode chegar até ±0,5°C em condições ideais. Essa precisão, somada à sua resolução de 0,02°C, garante medições detalhadas e confiáveis. Além disso, o MLX90614 é bastante econômico em termos de consumo de energia, consumindo cerca de 1 mA em operação, o que o torna perfeito para sistemas alimentados por bateria. Ele também se destaca pela facilidade de integração com microcontroladores, devido à sua interface I2C, bastante comum em sistemas embarcados.

Este sensor tem uma gama ampla de aplicações. Um dos usos mais populares é em termômetros infravermelhos para medir a temperatura corporal, como aqueles usados em hospitais ou para triagem de grandes públicos. Por não necessitar de contato com a pele, o risco de contaminação é reduzido, além de ser mais rápido e confortável para o paciente. No setor industrial, o MLX90614 é amplamente utilizado para monitorar a temperatura de processos de fabricação, ajudando a prevenir sobreaquecimento de máquinas e componentes. Também é útil para detectar falhas em equipamentos eletrônicos, já que pode medir a temperatura de circuitos e placas sem danificá-los. Além disso, em ambientes críticos, como data centers ou laboratórios, ele é fundamental para o controle ambiental, ajudando a manter a temperatura dentro dos parâmetros ideais.

No entanto, apesar de suas várias vantagens, o MLX90614 também apresenta algumas limitações. Por exemplo, ele pode ser afetado por condições ambientais adversas, como mudanças bruscas de temperatura ou superfícies altamente reflexivas, como metais polidos, que podem distorcer as medições. Sua precisão também depende da distância entre o sensor e o objeto, sendo ideal que essa distância esteja dentro de um intervalo recomendado de 5 cm a 30 cm para resultados mais confiáveis.

Apesar dessas limitações, o MLX90614 se destaca por sua facilidade de uso, precisão e eficiência energética, tornando-se uma escolha popular em diversas aplicações, desde

dispositivos portáteis até sistemas complexos de automação industrial. Sua capacidade de medir a temperatura de maneira remota e segura o coloca como uma ferramenta essencial em muitos campos, especialmente em um mundo cada vez mais voltado para a automação e segurança.

- **MQ-2**

O sensor MQ-2 é um dispositivo amplamente utilizado para detecção de gases em diversos ambientes, sendo uma das opções mais populares para projetos de automação e monitoramento ambiental. Com a capacidade de identificar gases como metano, monóxido de carbono, etanol e até fumaça, o MQ-2 se destaca pela sua versatilidade, podendo ser aplicado em sistemas de segurança, monitoramento de qualidade do ar e até em projetos de robótica.

Seu funcionamento se baseia em um princípio simples, mas eficaz. O sensor é composto por um material semicondutor de óxido metálico (SnO_2), cuja resistência elétrica varia quando entra em contato com certos gases. Quando esses gases entram em contato com o sensor, ocorre uma reação química que altera a condutividade do material, gerando um sinal elétrico que pode ser lido e interpretado por circuitos eletrônicos. Esse sinal é então convertido em uma medida de concentração do gás presente no ambiente.

No entanto, o sensor MQ-2 precisa de um tempo para atingir a temperatura de operação, o que significa que, ao ser ligado, ele precisa de alguns segundos para começar a dar respostas precisas. Esse processo é um ponto a ser considerado em aplicações que exigem uma detecção imediata. Além disso, como o sensor funciona com base na resistência elétrica, ele pode ser sensível a mudanças de temperatura e umidade, o que pode afetar a precisão das leituras. Em algumas situações, é necessário calibrar o sensor periodicamente para garantir que as medições estejam corretas.

Uma das grandes vantagens do MQ-2 é o seu baixo custo e a facilidade com que pode ser integrado a sistemas como o Arduino. Essa característica o torna uma escolha popular entre os hobbyistas e profissionais que desejam criar sistemas personalizados de detecção de gases ou de alarme de incêndio. Sua capacidade de detectar uma variedade de compostos químicos o torna bastante versátil, sendo ideal para ambientes internos, como casas e escritórios, mas também útil em algumas aplicações externas, desde que as condições ambientais sejam adequadas.

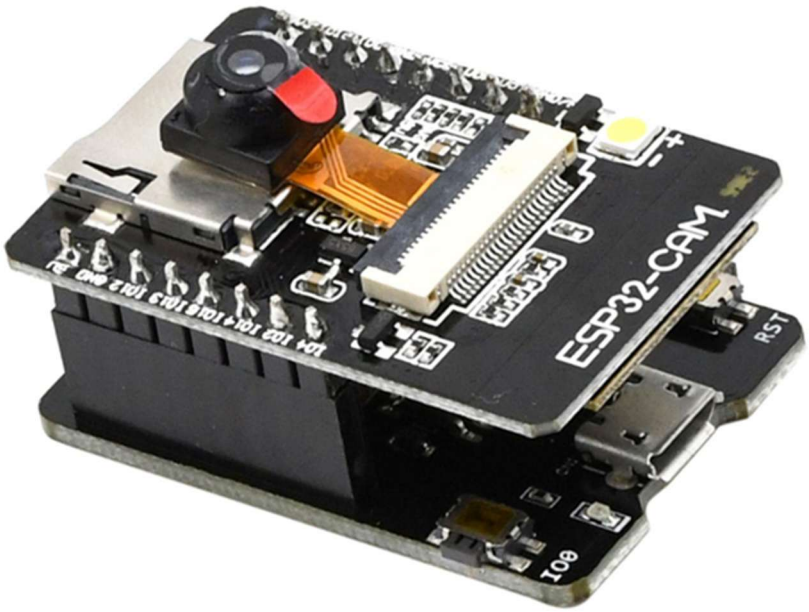
O sensor MQ-2 é frequentemente utilizado em sistemas de alarme de gás e fumaça, em que sua capacidade de detectar metano e monóxido de carbono, por exemplo, pode salvar vidas. Ele também é útil em projetos de automação residencial, onde o monitoramento constante da qualidade do ar pode ser integrado ao sistema de controle da casa inteligente. Além disso, em ambientes industriais, ele pode ser usado para detectar vazamentos de gás em áreas de risco, como cozinhas industriais ou instalações petroquímicas.

Entretanto, nem tudo é perfeito. O sensor MQ-2, apesar de seu baixo custo, exige cuidados adicionais, como calibração frequente, especialmente se for utilizado em ambientes com variações grandes de temperatura e umidade. Sua precisão pode ser comprometida se não for devidamente ajustado, e ele não é ideal para ambientes com concentrações muito altas de gases, o que pode saturá-lo e torná-lo incapaz de fornecer leituras precisas.

Ainda assim, sua acessibilidade e adaptabilidade fazem dele uma escolha sólida para muitos projetos. Se a aplicação exige um sensor que detecte a presença de gases como metano ou fumaça de forma simples e econômica, o MQ-2 pode ser uma excelente opção. Como qualquer sensor, ele exige que o usuário compreenda suas limitações, mas, quando bem utilizado, oferece um desempenho confiável e eficaz.

- **ESP32-CAM:**

Figura 16 - ESP32-CAM



Crédito: https://www.amazon.com.br/Aideepen-Bluetooth-ESP32-CAM-ESP32-CAM-MB-Arduino/dp/B08P2578LV?th=1

O ESP32-CAM é um módulo embarcado voltado para aplicações em Internet das Coisas (IoT) que combina o microcontrolador ESP32 com recursos de captura e transmissão de imagens. Ele é equipado com o processador dual-core Tensilica Xtensa LX6 de 32 bits, com frequência de operação de até 240 MHz, memória SRAM, suporte a flash externa, Wi-Fi e Bluetooth v4.2. Sua principal característica diferencial em relação a outros módulos da família ESP32 é a presença de uma interface dedicada a sensores de imagem, permitindo a integração com o sensor fotográfico OV2640, um dispositivo CMOS com resolução de até 2 megapixels (1600 × 1200), capaz de operar em diferentes modos de compressão, incluindo JPEG e YUV. Foi selecionado, a fim de realizar detecções apenas com imagens.

- **Confiabilidade estrutural:**

O ESP32-CAM foi projetado para operar em diferentes contextos de aplicação, apresentando estabilidade mesmo em ambientes sujeitos a variações de temperatura e condições adversas. Sua arquitetura otimizada garante desempenho confiável em tarefas contínuas, como monitoramento remoto e processamento de imagens em tempo real.

- **Gestão de energia:**

O módulo dispõe de modos de operação configuráveis que possibilitam o gerenciamento eficiente do consumo energético. Tal característica é essencial em aplicações móveis e em sistemas alimentados por bateria, uma vez que permite a conciliação entre baixo consumo e manutenção da capacidade de captura e transmissão de imagens.

- **Integração funcional:**

Com dimensões reduzidas, o ESP32-CAM agrega, em uma única solução, recursos de processamento, comunicação sem fio, captura de imagens, armazenamento local via microSD e gerenciamento de energia. Esta elevada integração reduz a complexidade de projetos eletrônicos e facilita a implementação de protótipos em aplicações de IoT.

- **Conectividade e processamento autônomo:**

O módulo é capaz de executar, de maneira independente, a captura, o processamento e a transmissão de imagens por meio da rede Wi-Fi, podendo ainda se comunicar com outros microcontroladores ou sistemas embarcados via interfaces UART, SPI ou I2C.

Essa flexibilidade amplia sua aplicabilidade em domínios como vigilância eletrônica, agricultura de precisão, reconhecimento facial e monitoramento ambiental.

Câmeras:

- **OV5640**

Oferece uma resolução significativamente superior, com 5 megapixels (2592x1944 pixels). Isso a torna ideal para aplicações nas quais a definição da imagem é crucial, como sistemas de monitoramento de vídeo, câmeras de celulares e outros dispositivos que requerem uma boa qualidade de imagem. A OV5640 também suporta a captura de vídeo em alta definição, mas sua taxa de quadros a 5 MP é limitada a 15 fps, o que pode não ser suficiente para algumas aplicações que exigem uma captura de vídeo fluida. Sua interface de comunicação, baseada no protocolo MIPI, é mais complexa, exigindo um hardware compatível, mas oferece uma qualidade de imagem muito superior, especialmente em relação à OV2640. A principal desvantagem da OV5640 é o maior consumo de energia, o que pode ser um fator limitante em projetos com restrições de autonomia.

- **OV7725**

É um sensor de baixa resolução, com apenas 0.3 megapixels (640x480 pixels), mas com uma taxa de quadros impressionante de até 60 fps a essa resolução. Isso a torna uma excelente escolha para aplicações em que a fluidez do vídeo é mais importante do que a definição da imagem, como sistemas de reconhecimento de movimento, visão computacional em tempo real e projetos que exigem captura rápida de imagens. A OV7725 é uma opção de baixo custo, com um consumo de energia reduzido, o que a torna adequada para dispositivos que precisam operar por longos períodos sem recarga. No entanto, sua resolução limitada impede que seja usada em projetos que exigem imagens detalhadas ou em condições de baixa iluminação.

- **OV2640:**

A OV2640, fabricada pela OmniVision, é uma câmera digital de aproximadamente 2 megapixels bastante conhecida no mundo dos sistemas embarcados. Ela se consolidou como uma das opções mais populares, principalmente por sua presença no módulo

ESP32-CAM, muito utilizado em projetos de IoT, monitoramento remoto e protótipos acadêmicos. O que a torna tão atrativa é o equilíbrio entre resolução, consumo reduzido de energia e a presença de um processador de imagem interno, capaz de realizar compressão JPEG e ajustes automáticos, o que simplifica enormemente a integração com microcontroladores de baixo poder de processamento.

Figura 17 - Câmera OV2640

Crédito: https://blog.arducam.com/ov2640/

- **Boa resolução:**

A OV2640 é capaz de capturar imagens em até 1600 × 1200 (UXGA), atingindo cerca de 2 megapixels de resolução. Essa qualidade é suficiente para muitas aplicações práticas, permitindo fotos nítidas e detalhadas em ambientes controlados. Entretanto, a câmera também pode operar em resoluções menores, como SVGA ou CIF, o que possibilita maior flexibilidade dependendo da necessidade do projeto.

- **Taxa de quadros:**

Em sua resolução máxima, a OV2640 alcança aproximadamente 15 quadros por segundo, um valor adequado para captura de imagens e até para transmissões mais lentas de vídeo. Quando configurada em resoluções menores, como SVGA, essa taxa pode chegar a 30 quadros por segundo, tornando-se mais indicada para aplicações que exigem fluidez no vídeo em tempo real.

- **Consumo de energia:**

OV2640 tem baixo consumo energético. Em operação típica, a câmera utiliza cerca de 60 mW de potência, o que a torna adequada para sistemas alimentados por bateria ou

energia solar. A tensão de operação situa-se entre 2,5 V e 3,0 V, permitindo uso direto em plataformas que trabalham com 3,3 V, como o ESP32.

Comunicação:

- **Bluetooth**

O Bluetooth, especialmente o Bluetooth Low Energy (BLE), é amplamente utilizado em dispositivos móveis e outros dispositivos próximos, devido ao seu baixo consumo de energia e facilidade de integração. Porém, em termos de alcance, o Bluetooth é significativamente mais limitado. Dependendo da classe do dispositivo, o alcance pode variar entre 10 e 100 metros, o que é adequado para comunicação de curto alcance, mas não serve para aplicações que exigem cobertura de longa distância.

Por outro lado, o Bluetooth oferece uma taxa de dados mais alta que o LoRa, com a capacidade de transferir até 3 Mbps em sua versão BLE. Isso o torna ideal para situações em que é necessário transmitir dados rapidamente ou em tempo real, como em sistemas de controle remoto, automação residencial e dispositivos vestíveis. Além disso, o Bluetooth tem baixa latência, o que significa que os dispositivos podem se comunicar de forma rápida e eficaz, o que é um benefício em aplicações interativas.

Embora o Bluetooth seja amplamente suportado em dispositivos móveis, suas limitações de alcance o tornam uma escolha inadequada para comunicação de longa distância. Se a sua aplicação requer apenas interação em ambientes locais ou em curto alcance, o Bluetooth é uma excelente escolha, mas para distâncias maiores, outras soluções seriam mais adequadas.

- **ESP-NOW**

O ESP-NOW, uma tecnologia do ESP32, oferece uma solução interessante para comunicação sem fio entre dispositivos próximos, utilizando o módulo de Wi-Fi do chip. O alcance do ESP-NOW pode variar entre 100 e 200 metros em ambientes abertos, o que o coloca em um ponto intermediário entre o Bluetooth e o LoRa. Embora não alcance a mesma distância do LoRa, ainda é uma boa opção para projetos de comunicação de médio alcance.

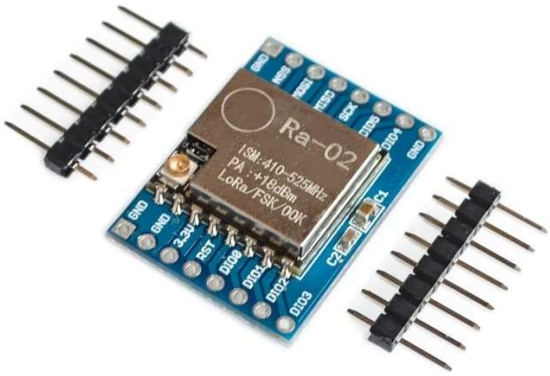
O grande diferencial do ESP-NOW é sua alta taxa de transmissão de dados, que pode chegar até 1 Mbps ou mais, tornando-o adequado para aplicações que exigem maior

largura de banda. Isso é vantajoso quando se precisa transmitir mais informações entre dispositivos, como em sistemas de monitoramento em tempo real ou comunicação rápida entre dispositivos em um ambiente local. Além disso, o ESP-NOW não necessita de infraestrutura Wi-Fi para funcionar, o que o torna prático para dispositivos móveis ou locais sem rede Wi-Fi.

No entanto, o consumo de energia do ESP32 é mais elevado do que o do LoRa, o que pode ser um ponto negativo em projetos que exigem longas durações com bateria limitada. Além disso, o alcance do ESP-NOW, embora seja maior que o do Bluetooth, ainda está distante do alcance do LoRa. Essa tecnologia é ideal para projetos que precisam de comunicação de médio alcance, como controle de dispositivos locais ou redes de sensores próximos.

- **Módulo LoRa (LoRa Alliance):**

Figura 18 - Módulo LoRa 410 MHz a 525 MHz



Crédito: https://arduinoecia.com.br/produto/modulo-lora-sx1278-433mhz-ra-02/?srsltid=AfmBOoqCG3IpjWp78y8qtqAkM3k8uOf_zaUQWtzLnZeYcEdDGLknkpcK

LoRa é uma tecnologia de comunicação sem fio que se destaca pelo longo alcance e baixo consumo de energia, permitindo a conexão de dispositivos da Internet das Coisas (IoT) por longas distâncias com pequenas quantidades de dados. O termo LoRa vem do inglês "Long Range" (longo alcance) e é usado em conjunto com o protocolo LoRaWAN para definir como os dispositivos se comunicam em uma rede. Essa tecnologia é amplamente utilizada em aplicações de cidades inteligentes, agricultura, rastreamento de ativos e monitoramento ambiental, em que é necessário que dispositivos funcionem por anos com baterias.

  - **Faixas de frequência:**

O protocolo LoRa opera em bandas de frequência sub-GHz, tipicamente nas faixas de 433 MHz, 868 MHz e 915 MHz, variando conforme a regulamentação regional e os

requisitos específicos da aplicação. Essas frequências sub-GHz são escolhidas por oferecerem melhor penetração em ambientes urbanos e maior alcance de comunicação em comparação às bandas de frequência mais altas.

- **Baixo consumo de energia:**

Permite que dispositivos funcionem por até 10 anos com baterias de baixa capacidade, graças à alta eficiência energética da tecnologia, tornando-a ideal para aplicações em locais remotos e de difícil acesso, onde a manutenção é limitada.

- **Baixo custo:**

Estabelece uma rede de comunicação de baixo custo e fácil implantação, dispensando a necessidade de infraestrutura complexa e permitindo rápida escalabilidade em ambientes diversos.

- **Longo alcance:**

O alcance da comunicação LoRa pode se estender por vários quilômetros, especialmente em áreas rurais ou com baixa densidade urbana, superando significativamente o alcance de outras tecnologias sem fio convencionais graças à sua modulação de espectro espalhado e alta sensibilidade de recepção.

Notificação/alerta:

- **SMS**

O SMS (Short Message System), ou Sistema de Mensagens Curtas, é uma forma de comunicação simples e direta que permite o envio de mensagens curtas de texto para um número de telefone celular. A principal vantagem do SMS é sua rapidez. As mensagens são entregues quase instantaneamente, e as taxas de leitura são altíssimas – cerca de 90% das mensagens são abertas dentro de poucos minutos após o envio. Isso faz do SMS a escolha ideal para sistemas de alerta em situações de emergência ou que requerem uma resposta imediata. Se a urgência é a prioridade, o SMS se destaca por ser um canal de comunicação muito eficiente.

Além disso, o SMS não depende de internet. Ele funciona em qualquer telefone celular, seja ele básico ou smartphone, tornando-o acessível a uma vasta gama de usuários, especialmente em regiões com infraestrutura digital limitada. Isso também o torna um método de comunicação universal, atingindo praticamente qualquer pessoa que tenha um celular.

No entanto, essa rapidez e acessibilidade vêm com algumas limitações. O espaço para o conteúdo é restrito – uma mensagem de SMS não pode ultrapassar 160 caracteres. Isso obriga quem envia a ser direto e conciso, o que pode ser uma vantagem em termos de clareza, mas também limita a quantidade de informações que podem ser compartilhadas em uma única mensagem. Em situações que exigem mais explicações ou informações detalhadas, o SMS pode não ser o melhor canal.

Além disso, o custo do SMS pode ser um fator a ser considerado, especialmente em alertas de grande escala. Embora existam pacotes empresariais para envio de SMS em massa, o custo por mensagem pode ser elevado, o que torna o SMS uma opção mais cara para sistemas de alerta que necessitam enviar muitas notificações.

Por fim, a segurança é outro ponto a ser observado. O SMS não é o meio mais seguro para a transmissão de dados sensíveis, já que está sujeito a fraudes como o *smishing* (phishing via SMS), que pode comprometer a privacidade do usuário.

- **E-mail**

O e-mail, por sua vez, oferece uma comunicação mais detalhada e flexível. Ele permite a inclusão de imagens, links, arquivos anexos e formatação rica de texto, o que o torna uma opção muito mais robusta quando o objetivo é fornecer informações complexas ou explicações mais aprofundadas. Se a notificação requer um conteúdo mais extenso ou personalizado, o e-mail é o meio ideal. A possibilidade de incluir gráficos, tabelas e outros recursos visuais é um diferencial importante, principalmente em contextos corporativos ou em sistemas que necessitam de relatórios detalhados.

Outro grande benefício do e-mail é seu custo, que é consideravelmente baixo em comparação com o SMS. Para sistemas de alerta que precisam enviar notificações em grande volume, o e-mail se torna uma opção muito mais econômica. Não há custos adicionais para o envio de múltiplas mensagens, o que o torna altamente escalável, especialmente para empresas ou organizações que precisam atingir um grande número de pessoas ao mesmo tempo.

No entanto, o e-mail também possui suas desvantagens. A taxa de leitura de e-mails é geralmente mais baixa do que a do SMS, especialmente para mensagens não urgentes. Muitos usuários simplesmente ignoram notificações de e-mail ou não verificam suas caixas de entrada com a mesma frequência que as mensagens de texto. Isso pode levar a um atraso na leitura das mensagens, o que é problemático em contextos em que a

rapidez é essencial. Além disso, o e-mail depende de uma conexão com a internet, o que pode ser uma limitação em áreas com baixa conectividade.

A segurança também é um ponto relevante quando se trata de e-mails. Embora o e-mail ofereça opções de proteção, como autenticação de dois fatores e criptografia, ele ainda é vulnerável a ataques de phishing e fraudes, que podem colocar em risco a segurança das informações e dos usuários. Apesar dessas opções de segurança, o e-mail continua sendo um canal mais suscetível a tentativas de fraude do que o SMS.

Em relação à usabilidade, o e-mail oferece uma maior flexibilidade em termos de personalização, permitindo que o remetente configure a mensagem de acordo com o perfil do destinatário, o que não é possível com o SMS.

- **Telegram:**

Figura 19 - Logo do softwareTelegram

Crédito: https://pt.wikipedia.org/wiki/Telegram

O Telegram é um serviço de mensagens instantâneas baseado em nuvem disponível em diversas plataformas, incluindo smartphones, tablets e computadores. Permite chamadas de vídeo, envio de mensagens, fotos, vídeos, autocolantes e arquivos de qualquer tipo. O Telegram oferece a opção de criptografia de ponta a ponta. Enquanto os clientes do Telegram têm código aberto, seus servidores são proprietários. A plataforma também disponibiliza APIs para desenvolvedores independentes. Fundado em 2013 pelos irmãos Nikolai e Pavel Durov, os mesmos fundadores do VK, a maior rede social da Rússia, o Telegram é uma entidade independente, não associada ao VK. Atualmente tem sede em Dubai, Emirados Árabes Unidos.

Fonte de alimentação:

Foram analisadas as opções:

- Bateria;

- Rede elétrica.

Pesquisas conduzidas pela equipe mostraram que inicialmente o custo para instalação de um sistema de alimentação solar era alto em relação ao de utilizar alimentação por cabos elétricos provindos da rede. Em contrapartida, não há viabilidade em utilizar um sistema de alimentação por conexão de cabos para um sistema consideravelmente afastado da rede elétrica, por exemplo. Com isso, escolheu-se utilizar somente a bateria.

Bateria

Figura 20 - Bateria Li-Po de 7,4V

Crédito: https://www.lojauab.com.br/bateria-lipo-74v-450-mah-aep

A bateria Li-Po é um tipo de bateria de polímero de lítio (lítio-polietileno) amplamente utilizada em várias aplicações, como drones, modelos de controle remoto, carros elétricos e até aparelhos portáteis.

- **Alta densidade de energia:**

As baterias Li-Po têm uma excelente densidade de energia, o que significa que conseguem armazenar mais energia em um espaço mais compacto, quando comparadas com baterias como as de níquel-cádmio (NiCd) ou níquel-metal-hidreto (NiMH).

- **Leveza:**

As baterias Li-Po são mais leves do que outras químicas, como as de lítio-íon (Li-ion) ou chumbo-ácido, o que ajuda a reduzir o peso total dos dispositivos em que são usadas. Essa característica é essencial para aparelhos que exigem alta portabilidade e desempenho, como drones e modelos RC.

- **Taxa de descarga alta:**

As baterias Li-Po podem fornecer grandes quantidades de corrente rapidamente, o que é crucial para aplicações que exigem picos de energia instantâneos, como motores de

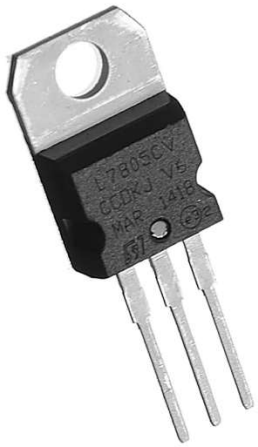
alta performance, por exemplo. São ideais para dispositivos que demandam alto desempenho de potência.

- **Tamanho e flexibilidade:**

Ao contrário de baterias tradicionais como as de NiMH, as baterias de Li-Po podem ser fabricadas em formas mais finas e customizáveis, permitindo maior flexibilidade no design dos dispositivos.

Regulador de tensão 7805:

Figura 21 - Regulador de tensão L7805CV



Crédito: https://www.marinostore.com/componentes/lm7805cv-regulador-de-tensao-5v?srsltid=AfmBOoqULb_Z7zeD_1Js4o34DVhywLcmuleXPqHgPwefai7FuTikapJV

Utilizou-se o regulador de tensão 7805 para adequar os níveis de tensão da bateria (7,4 V) para o valor nominal de operação do ESP32 e ESP32-CAM (5 V). Exceder a tensão de operação pode danificar os componentes.

- **Design robusto:**

O regulador 7805 possui construção robusta e estável, operando de forma segura mesmo com variações na tensão de entrada. Suas proteções internas contra sobreaquecimento e curto-circuito aumentam a confiabilidade. O encapsulamento TO-220 permite boa dissipação térmica, especialmente com dissipador.

- **Especificações elétricas:**

Ele fornece tensão fixa de 5 V com corrente de até cerca de 1,5 A. Exige tensão de entrada entre aproximadamente 7 V e 35 V devido ao dropout típico de 2 V. Apresenta boa estabilidade, baixa variação sob carga e saída com pouco ruído.

- **Facilidade de integração:**

O 7805 integra circuitos internos de proteção e referência, reduzindo a necessidade de componentes adicionais. Seu uso é simples graças aos três pinos: entrada, terra e saída. Apenas alguns capacitores são necessários para garantir estabilidade e reduzir ruídos.

- **Variedade de aplicações:**

É amplamente utilizado para alimentar microcontroladores, sensores e módulos eletrônicos que exigem 5 V estáveis. Pode ser empregado em fontes lineares ou integrado diretamente em PCBs. Seu baixo custo e confiabilidade o tornam ideal para projetos acadêmicos e profissionais.

Servo Motor MG995:

Figura 22 - Servo Motor MG995 com acessórios

Crédito: https://makerselectronics.com/product/servo-motor-half-metal-gear-mg995-towerpro-continuous-rotation/

O servo-motor MG995 é um atuador eletromecânico amplamente utilizado em sistemas de automação, robótica e projetos de eletrônica embarcada. Trata-se de um motor DC com engrenagens metálicas internas, sensor de posição (potenciômetro) e circuito de controle integrado, capaz de realizar movimentos angulares de forma precisa e controlada. Suas principais características incluem torque elevado, resposta rápida, confiabilidade, resistência mecânica e ampla aplicabilidade em protótipos e sistemas industriais de pequeno porte.

- **Construção robusta:**

O MG995 é projetado com engrenagens metálicas de alta durabilidade, o que lhe confere maior resistência ao desgaste e maior estabilidade em aplicações que exigem força contínua. Essa robustez permite seu uso em ambientes com vibrações e cargas variáveis sem comprometer o desempenho.

- **Torque elevado e precisão:**

Este servo é capaz de fornecer torque significativo, em torno de 9–10 kgf·cm a 6 V, possibilitando a movimentação de estruturas relativamente pesadas em projetos de robótica. Além disso, o sistema de realimentação por potenciômetro garante precisão no posicionamento angular dentro do seu limite de operação (aproximadamente 0° a 180°).

- **Alta integração:**

O MG995 integra, em um único módulo, o motor DC, as engrenagens redutoras, o circuito de controle eletrônico e o sensor de posição. Essa integração facilita sua utilização, necessitando apenas de uma alimentação contínua e um sinal PWM para controle de posição.

- **Versatilidade de aplicações:**

O servo-motor MG995 pode ser utilizado em braços robóticos, sistemas de direção de veículos em miniatura, acionamento de mecanismos articulados, aeromodelos e projetos educacionais. Sua compatibilidade com microcontroladores como Arduino e ESP32 o torna especialmente útil em protótipos de automação e experimentos de eletrônica e mecatrônica.

Google API Vision (Google Cloud)

Figura 23 - Logo Google Cloud

Crédito: https://safetec.com.br/cloud-computing/o-que-e-google-cloud-platform/

O Google Cloud é um conjunto de serviços de computação em nuvem que oferece a infraestrutura do Google (como computação, armazenamento, análise de dados e machine learning) como um serviço para empresas e desenvolvedores. Ele permite que

as empresas utilizem a infraestrutura global do Google sem a necessidade de ter seus próprios data centers, transformando gastos de capital em despesas operacionais flexíveis e permitindo o pagamento pelo uso.

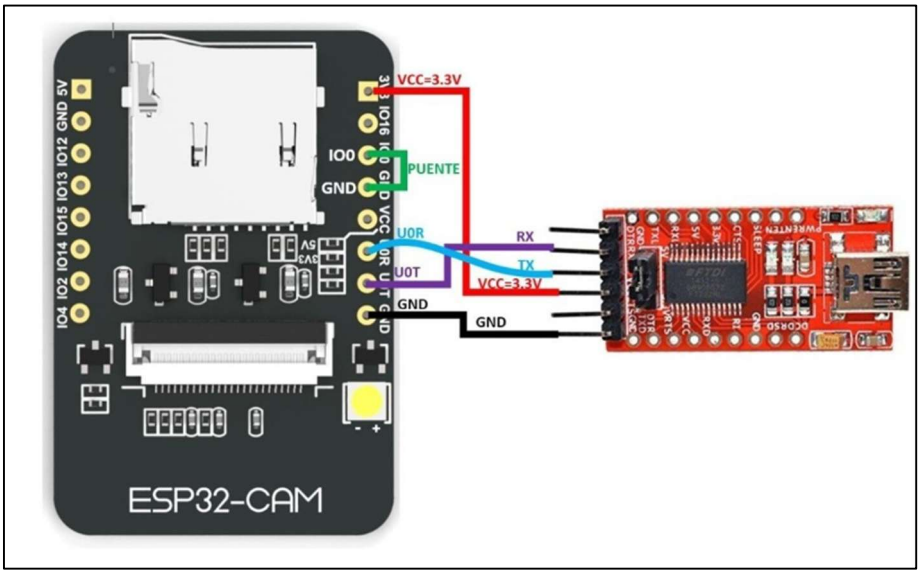
Figura 24 - Logo Google Vision API

Crédito: https://www.simpleocr.com/product/google-cloud-vision-api/

Na plataforma, foi utilizado o Google Vision API, serviço que utiliza inteligência artificial e um banco de dados para identificar entidades em imagens compiladas ao serviço.

Ao concluir-se a abordagem do tema do projeto e identificar a tecnologia principal do protótipo, a I.A. para detecção em imagens, foi empreendida uma pesquisa em diversas fontes acerca do funcionamento do módulo ESP32-CAM, bem como o serviço de APIs do Google Cloud, plataforma desenvolvida pela Google para a criação de projetos, com serviços fornecidos pela própria e pela comunidade. Após a realização de alguns testes, foi definida a montagem mais eficaz para o circuito. Para a montagem, o ESP32-CAM é conectado ao computador de forma independente. A imagem abaixo ilustra métodos de conexão do ESP32-CAM.

Figura 25 - ESP32-CAM conectado a módulo FTDI



Crédito: https://forum.dronebotworkshop.com/esp32-esp8266/esp32-cam-mb-and-esp32-cam/

No caso, utilizou-se um shield próprio do ESP32-CAM para garantir a entrada de sinal USB, ao invés do módulo FTDI, que faz necessário a conexão por jumpers entre ele e o módulo câmera, o que polui visualmente o circuito. A imagem abaixo mostram o shield:

Figura 26 - Shield ESP32-CAM

Crédito: https://www.makerhero.com/produto/shield-de-programacao-esp32-cam-mb/?srsltid=AfmBOoqBgL05UvuJbuUS2QbH3JeC1pHV6TMGd2VQJWKgLp4ij_w1Q3zx

É elaborada uma tabela dos gastos para a montagem do projeto.

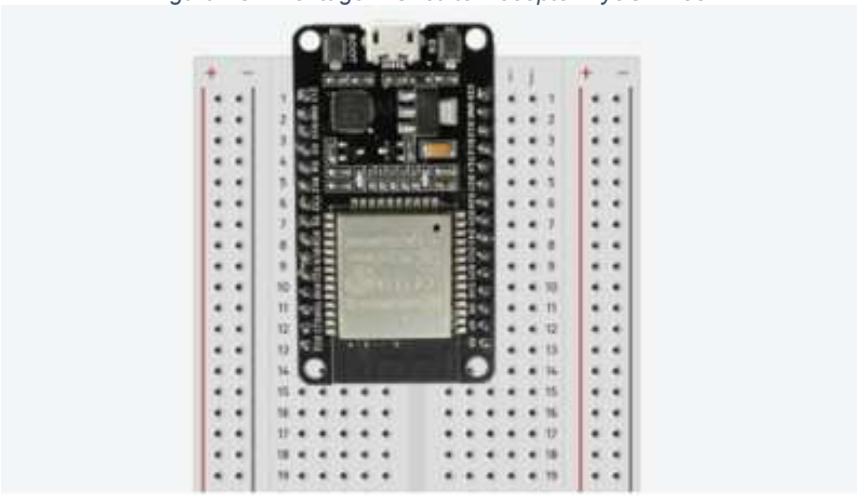
Figura 27 - Tabela de gastos EyeOnTree

Componente	Qtde.	Valor Unit. (R$)	Frete/taxa (R$)	Total (R$)
ESP32-CAM	1	12,6	39,39	51,99
ESP32 + cabo	1	89,99	0	89,99
Câmera OV5640	1	42,71	53,85	96,56
Servo motor MG995	1	36,78	0	36,78
Bateria Li-Po 7,4V	1	78,95	26,07	105,02
Protoboard 400 furos	1	14,25	0	14,25
Cabo tipo C (2x)	1	16,52	0	16,52
TOTAL FINAL (R$)				**411,11**

Crédito: de autoria própria utilizando o Excel

Referente à montagem do circuito do ESP32, a placa é conectada à protoboard, visando obter um maior número de pontos de conexão, conforme ilustrado na imagem abaixo:

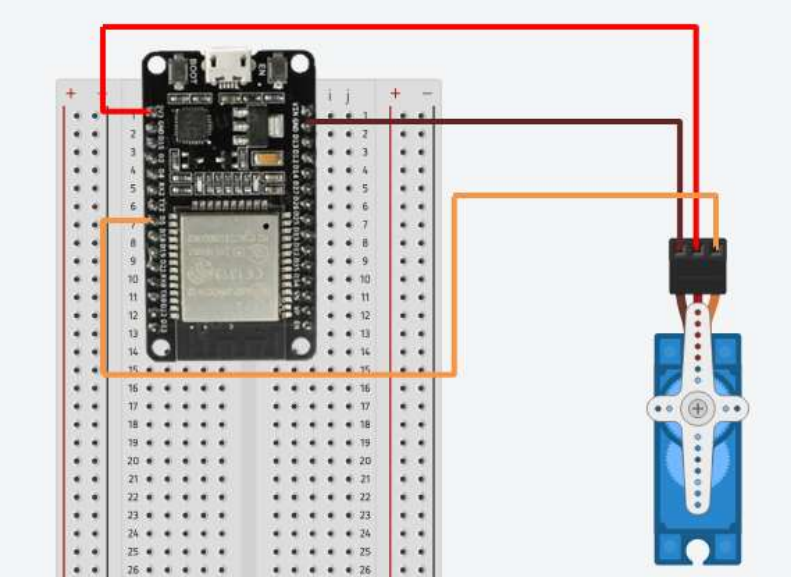
Figura 28 - Montagem Circuito Receptor EyeOnTree



Crédito: de autoria própria utilizando o PowerPoint e o TinkerCad

Em seguida, realiza-se a conexão do servo-motor. O terminal GND do servo-motor (de cor marrom) foi conectado ao pino GND do ESP32; O terminal de alimentação do motor (de cor vermelha) foi conectado ao pino 3V3 (3,3 Volts) do ESP32. Por fim, o terminal para envio de informações do servo-motor (de cor laranja) foi conectado ao pino D5, vide a imagem a seguir:

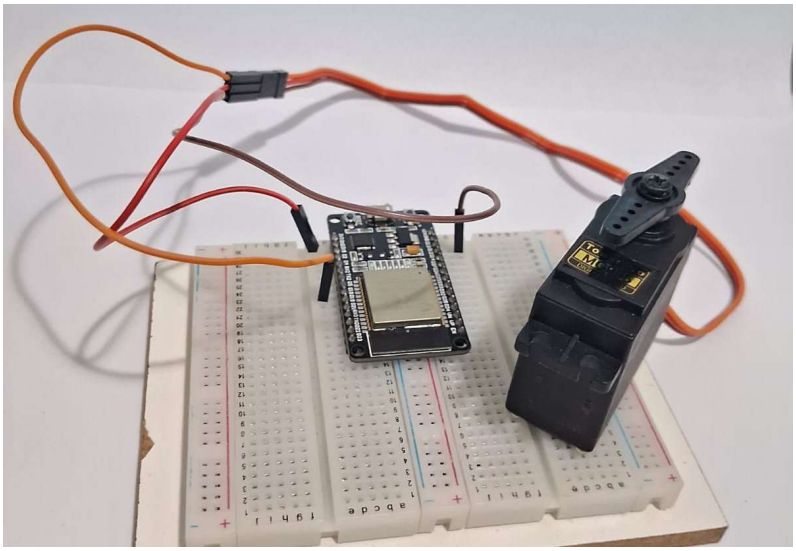
Figura 29 - Montagem Circuito Receptor EyeOnTree



Crédito: de autoria própria utilizando o PowerPoint e o TinkerCad

Visualmente, esse circuito é visto da seguinte forma:

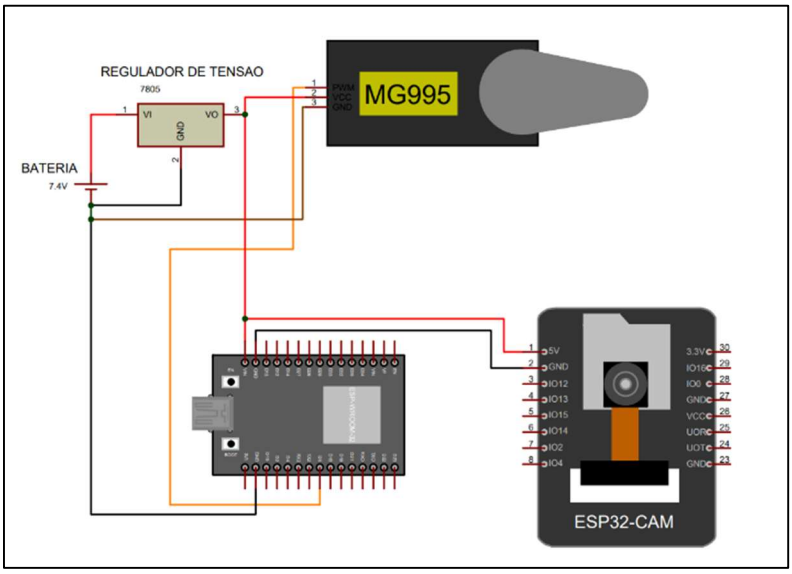
Figura 30 - Montagem Circuito Receptor EyeOnTree



Crédito: de autoria própria

Após juntar ambos os circuitos, o sistema fica assim:

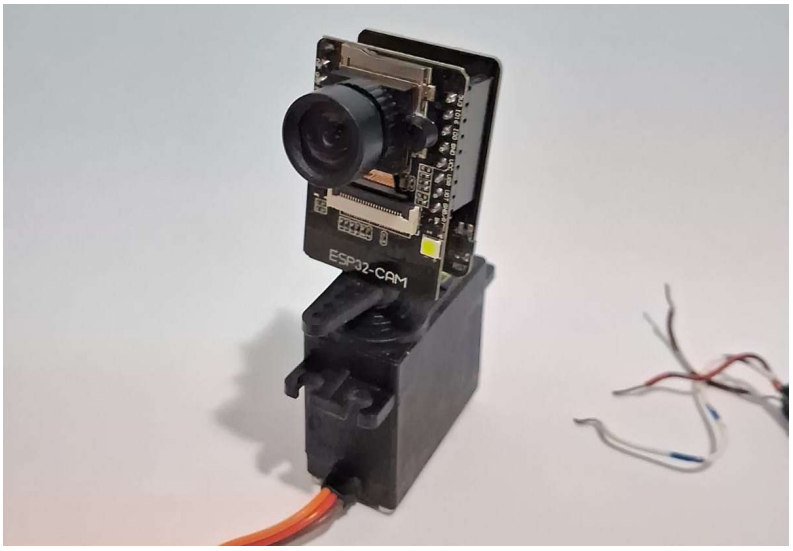
Figura 31 - Montagem Circuito Completo EyeOnTree



Crédito: de autoria própria utilizando o Proteus VSI

Ambos os circuitos finalmente montados, o ESP32-CAM será sobreposto ao circuito do ESP32:

Figura 32 - Circuito ESP32-CAM sobreposto ao circuito receptor

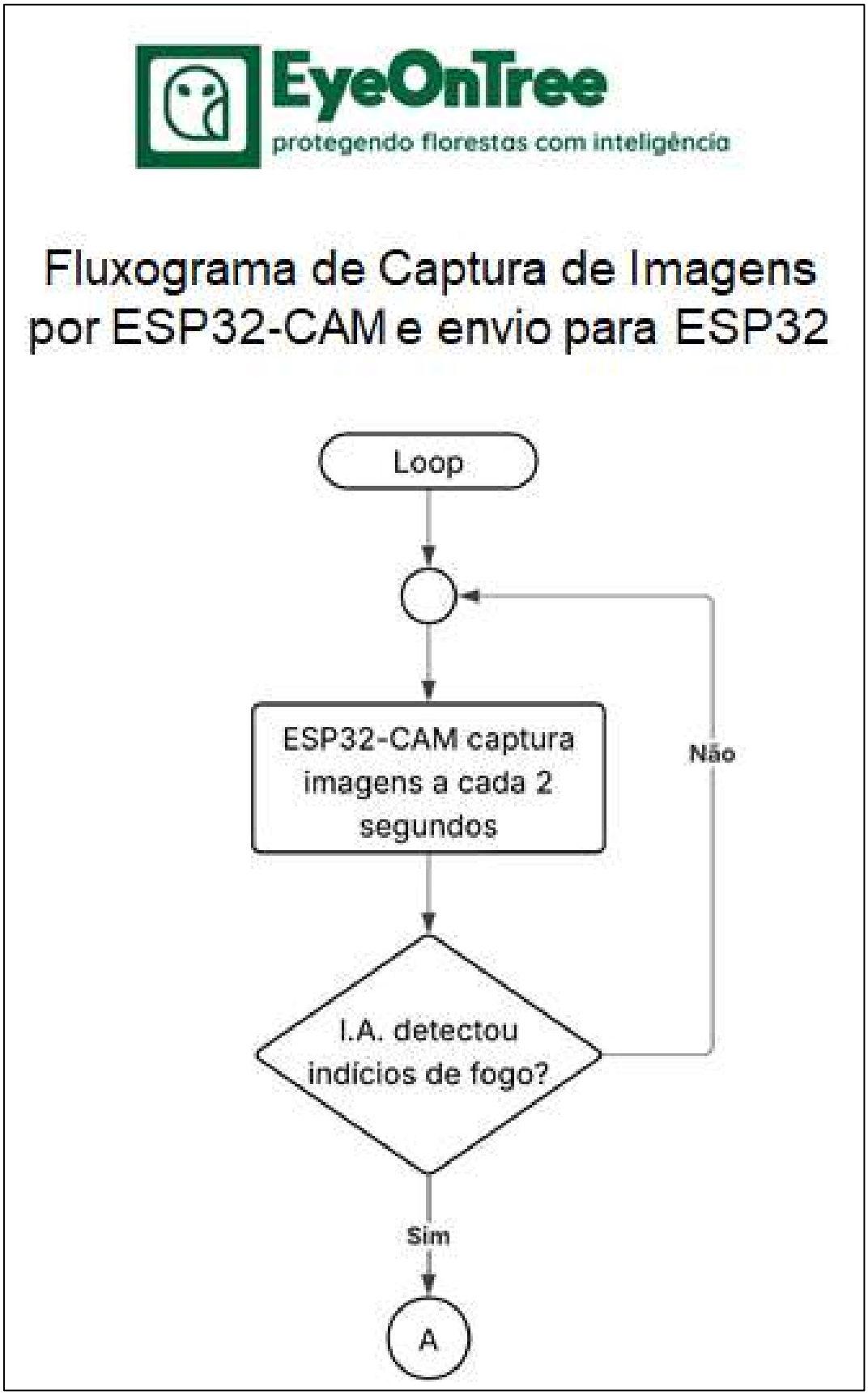


Crédito: de autoria própria

Para explicar o funcionamento do circuito e da lógica de programação, foi desenvolvido um fluxograma para facilitar a compreensão de maneira clara e direta. Durante a fase de montagem do projeto, foi identificada a necessidade de criar dois códigos distintos: um para a captura de imagens e outro para enviar um alerta condicional para o proprietário.

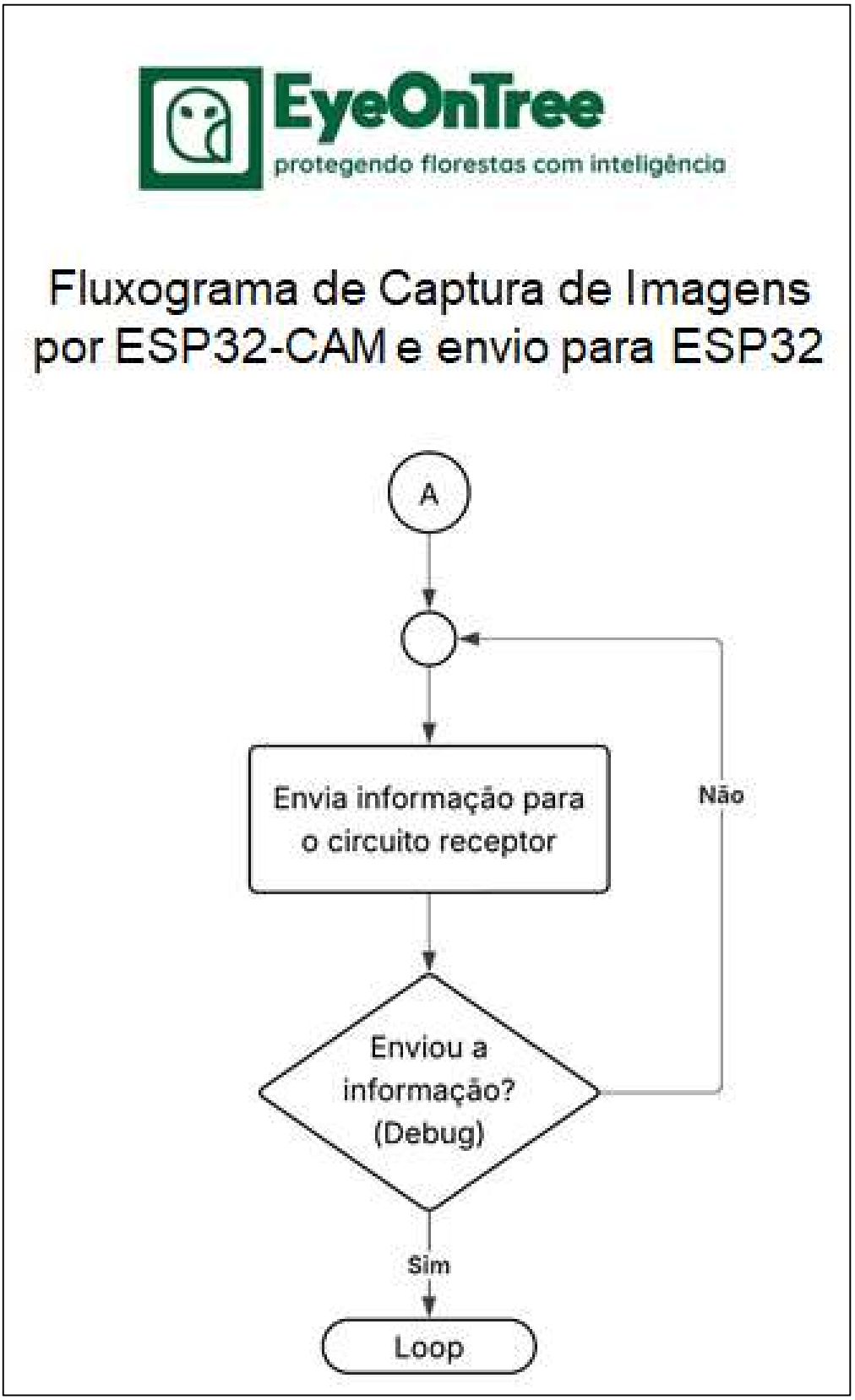
A seguir, a análise do funcionamento do sistema do ESP32-CAM:

Figura 33 - Fluxograma de Captura de Imagens por ESP32-CAM e envio para ESP32



Crédito: de autoria própria utilizando o Lucidchart e PowerPoint

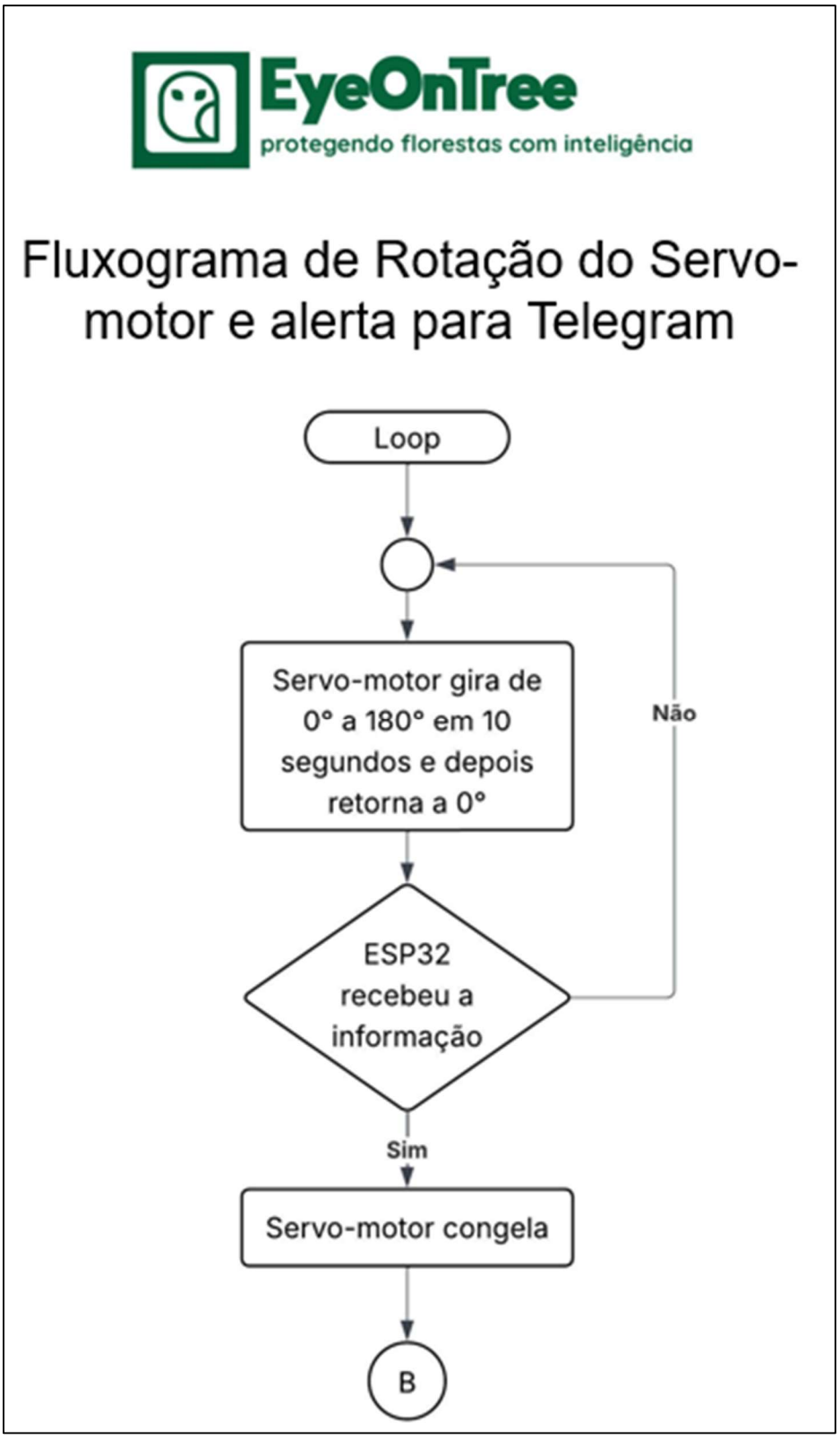
Figura 34 - Fluxograma de Captura de Imagens por ESP32-CAM e envio para ESP32



Crédito: de autoria própria utilizando o Lucidchart e PowerPoint

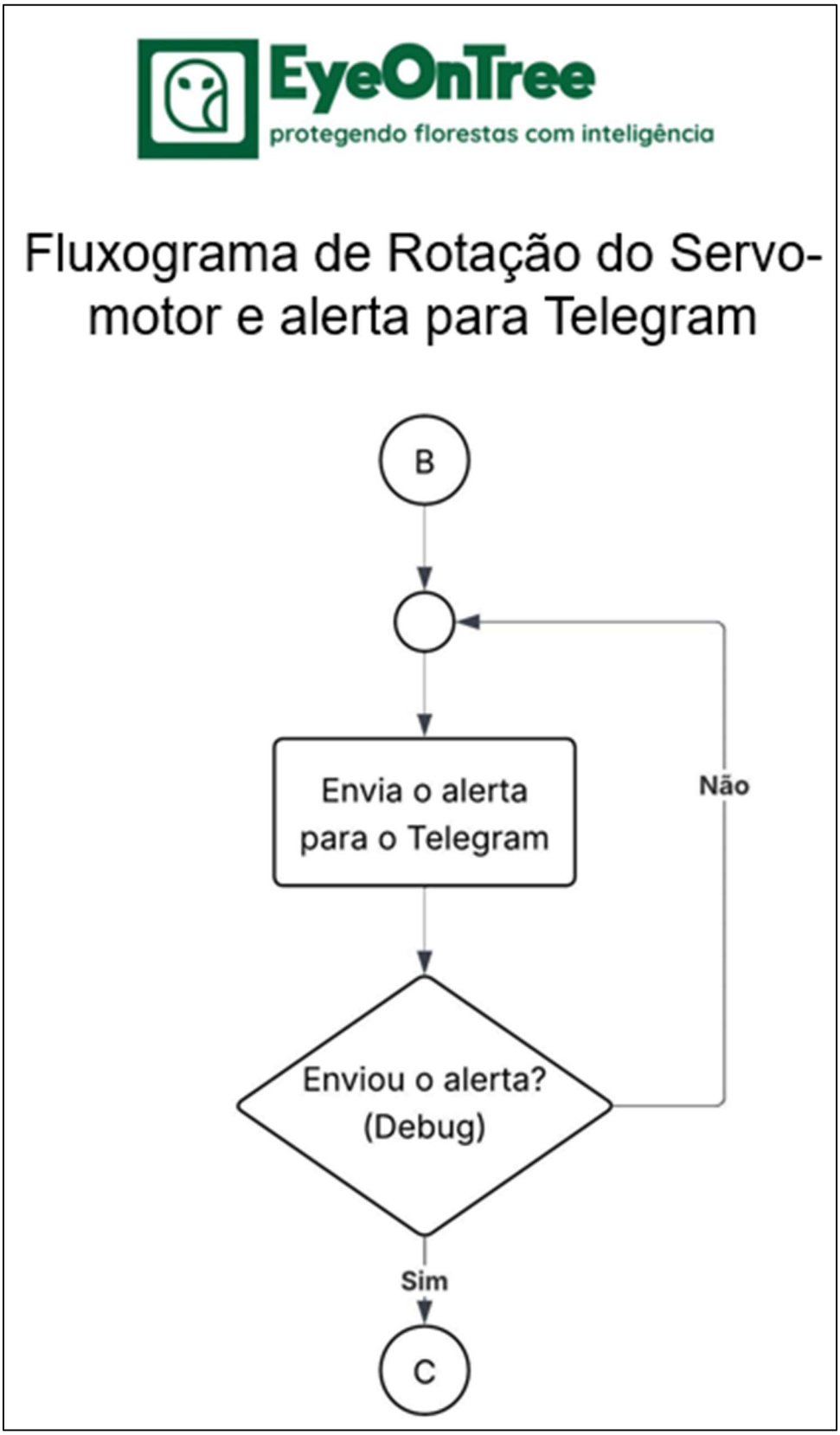
Agora, o funcionamento do sistema do ESP32:

Figura 35 - Fluxograma de Rotação do Servo-motor e alerta para Telegram



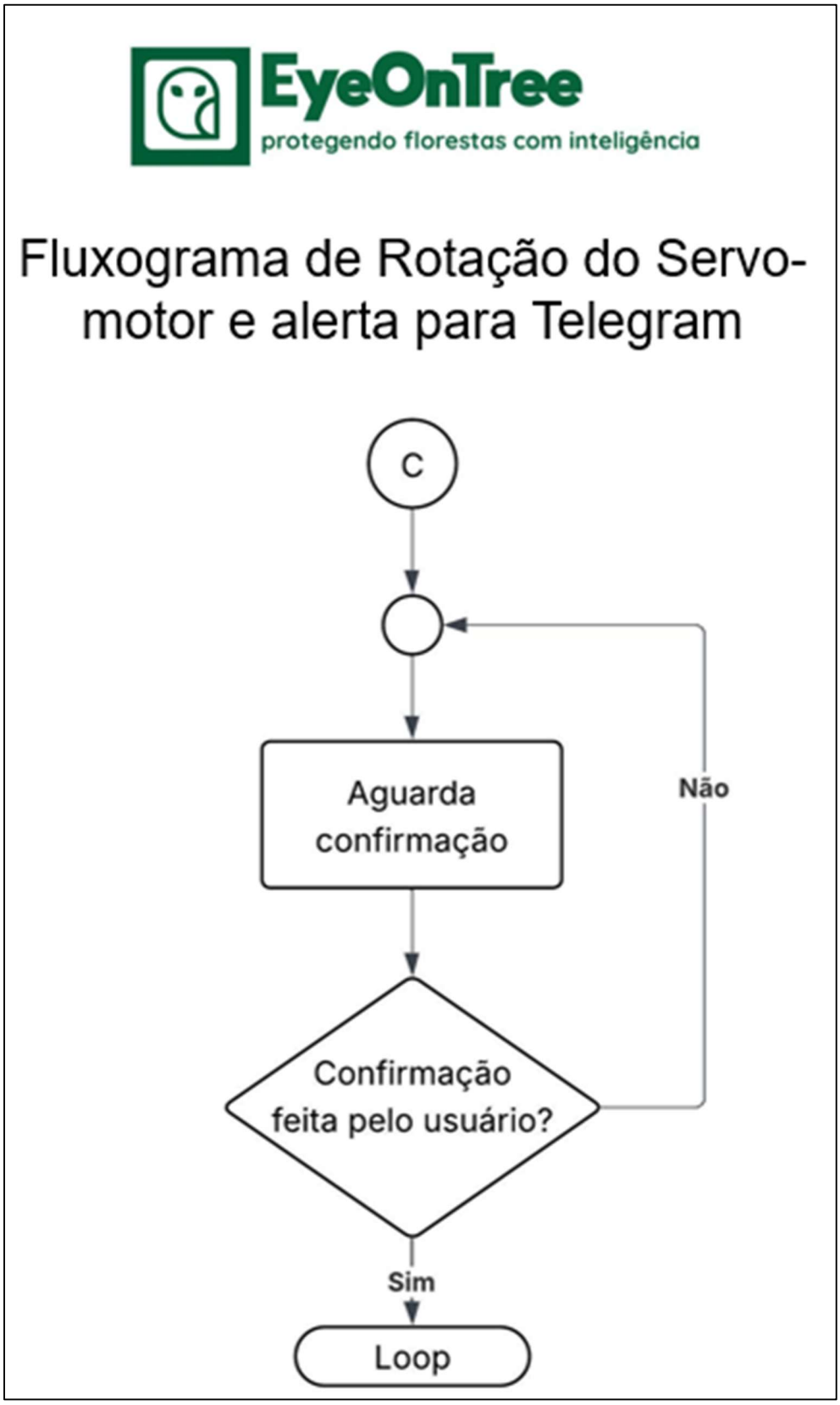
Crédito: de autoria própria utilizando o Lucidchart e PowerPoint

Figura 36 - Fluxograma de Rotação do Servo-motor e alerta para Telegram



Crédito: de autoria própria utilizando o Lucidchart e PowerPoint

Figura 37 - Fluxograma de Rotação do Servo-motor e alerta para Telegram

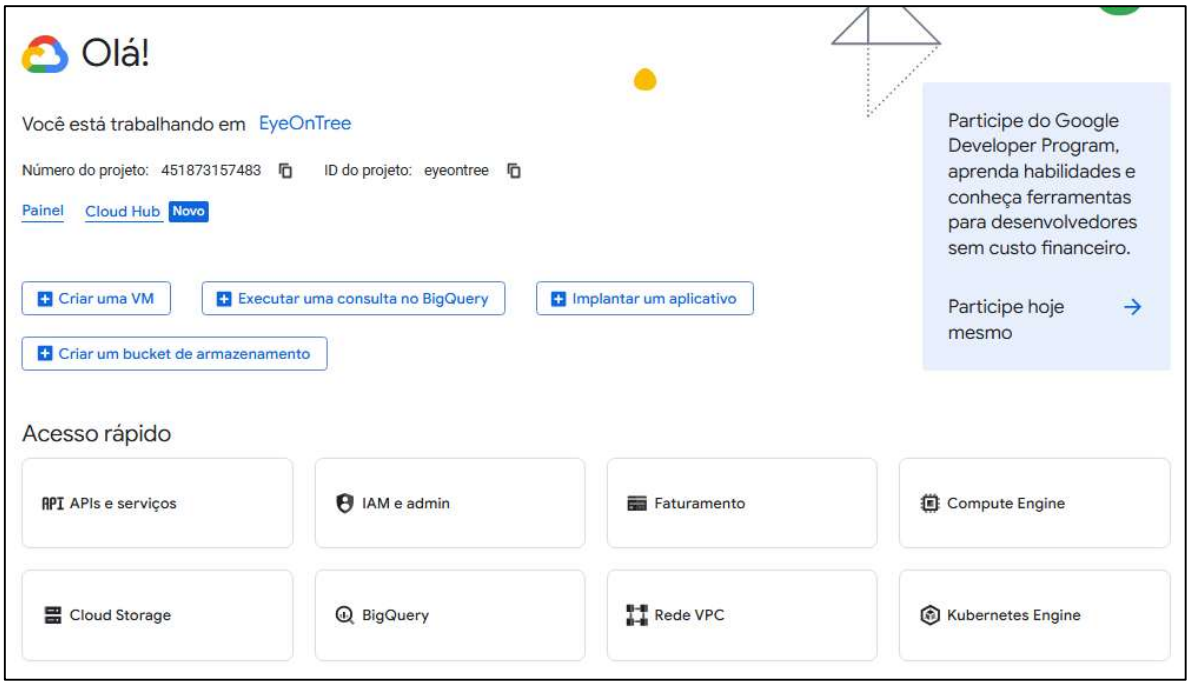


Crédito: de autoria própria utilizando o Lucidchart e PowerPoint

Criação da chave da API Google Vision

Primeiramente, cria-se um projeto na plataforma Google Cloud.

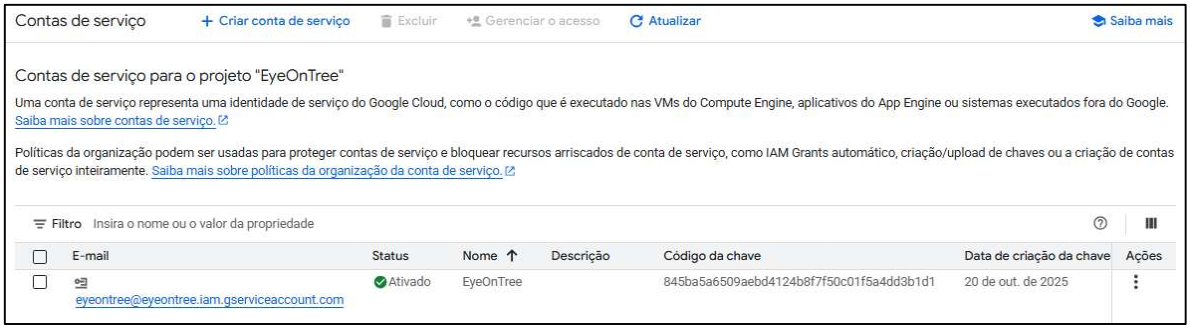
Figura 38 - Tela inicial Google Vision API



Crédito: de autoria própria utilizando o Google Vision API

Após adicionar uma conta de faturamento, são criados dados como uma chave de acesso e um número de identificação, que serão utilizados posteriormente na programação para interligar o serviço e o sistema. Vale ressaltar que para o feito não houve gasto financeiro.

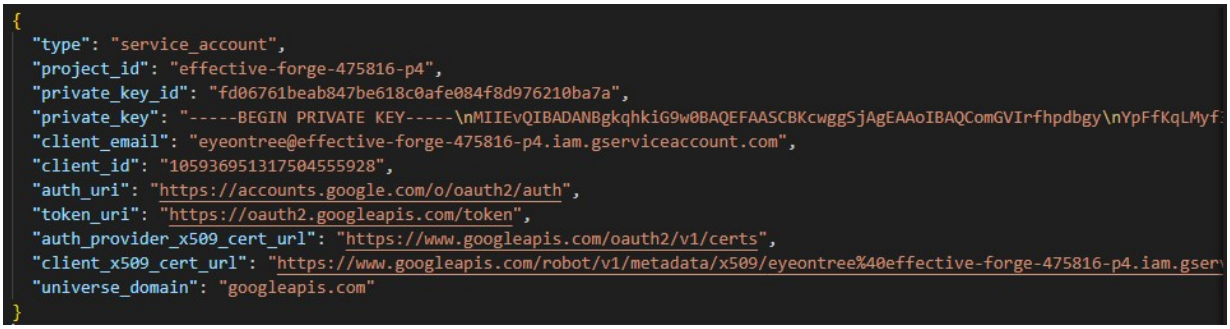
Figura 39 - Conta de serviço e chave de acesso EyeOnTree



Crédito: de autoria própria utilizando o Google Vision API

A partir do Visual Studio Code, plataforma de programação, é possível visualizar a chave da API, dentre outras configurações, utilizando a linguagem de programação “Node.js” (JSON).

Figura 40 - Chave API EyeOnTree



Crédito: de autoria própria utilizando o Visual Studio Code

São validadas as versões do Node.js e se essas conseguem se conectar com o serviço de APIs do Google.

Figura 41 - Validação das versões do Node.js

Crédito: de autoria própria utilizando o Visual Studio Code

Por fim, o sistema é testado com uma imagem de teste armazenada em uma pasta do mesmo diretório. O código deve compilar a imagem, enviá-la à API, e esta deve analisá-la, retornando os valores para o Node.js.

Figura 42 - Teste Node.js

```
'use strict';

function main() {
  // [START vision_quickstart]
  async function quickstart() {
    // Imports the Google Cloud client library
    const vision = require('@google-cloud/vision');

    // Creates a client
    const client = new vision.ImageAnnotatorClient();

    // Performs label detection on the image file
    const [result] = await client.labelDetection('./resources/test.jpg');
    const labels = result.labelAnnotations;
    console.log('Labels:');
    labels.forEach(label =>{ console.log(label)});
  }
  quickstart();
  // [END vision_quickstart]
}

process.on('unhandledRejection', err => {
  console.error(err.message);
  process.exitCode = 1;
});

main(...process.argv.slice(2));
```

Crédito: de autoria própria utilizando o Visual Studio Code

Logo, baseado no teste, um servidor é criado para que o serviço possa se comunicar com o código em C++, explicado mais abaixo. De forma direta, eles não se conectam, então o servidor faz essa ponte entre os componentes.

Figura 43 - Servidor Node.js

```
var fs = require('fs');
const http = require('http');
const server = http.createServer();
const filePath = './resources/test.jpeg';

server.on('request', (request, response)=>{
    if(request.method == 'POST' && request.url === "/imageUpdate"){

        var ImageFile = fs.createWriteStream(filePath);
        request.on('data', function(data){
            ImageFile.write(data);
        });

        request.on('end',async function(){
            ImageFile.end();
            const labels = await labelAPI();
            response.writeHead(200, {'Content-Type' : 'application/json'});
            response.end(JSON.stringify(labels));
        });

    }else{
        console.log("error");
        response.writeHead(405, {'Content-Type' : 'text/plain'});
        response.end();
    }
});
```

Crédito: de autoria própria utilizando o Visual Studio Code

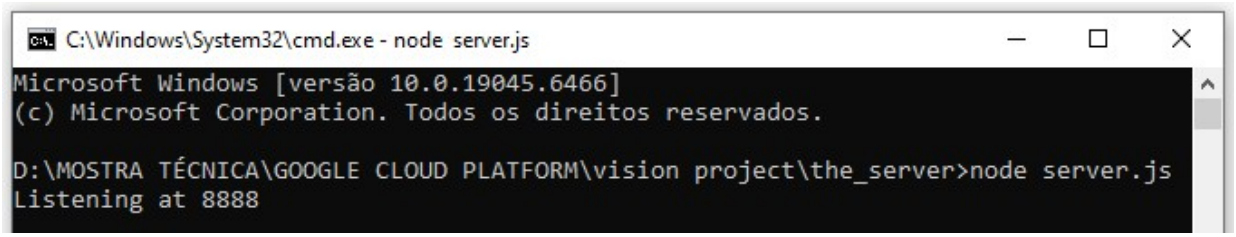
Figura 44 - Servidor Node.js

```
async function labelAPI() {
  var o = [];
  const vision = require('@google-cloud/vision');

  // 🔑 caminho direto da chave JSON
  const client = new vision.ImageAnnotatorClient({
    keyFilename: "C:\\GOOGLE CLOUD PLATFORM\\vision project\\the_key.json"
  });

  const [result] = await client.labelDetection(filePath);
  const labels = result.labelAnnotations;

  labels.forEach(label => {
    o.push({description: label.description, score: label.score});
  });
  return o;
}

const port = 8888;
server.listen(port);
console.log(`Listening at ${port}`);
```

Crédito: de autoria própria utilizando o Visual Studio Code

Agora, ao abrir o Prompt de Comando (CMD) dentro da pasta onde se localiza o servidor, enviando “node server.js”, deve-se obter uma resposta como “Listening at 8888”, o que significa que a conexão foi bem feita e o sistema está funcionando com êxito.

Figura 45 – Confirmação de Funcionamento do Servidor



Crédito: de autoria própria utilizando o CMD

Considerando a criação de dois circuitos, faz-se necessário o desenvolvimento de dois códigos, ambos divididos em partes e explicados a seguir. O primeiro código é responsável pela captura de imagens e envio pelo ESP32-CAM (Circuito transmissor), enquanto o segundo código se trata do recebimento das informações do ESP32-CAM pelo ESP32 (Circuito receptor) e envio de alerta desse para o Telegram do proprietário.

Circuito transmissor (ESP32-CAM)

Pré-processamento:

Efetua-se a inclusão de bibliotecas:

- esp_camera (por Espressif Systems)
- WiFi (por Arduino.cc)
- HTTPClient (por Adrian McEwen)
- ArduinoJson (por Benoit Blanchon)

Figura 46 - Programação da Captura de Imagens do ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
```

Crédito: Autoria própria Utilizando o Arduino IDE

Determina-se a numeração dos terminais do ESP32-CAM.

Figura 47 - Programação da Captura de Imagens do ESP32-CAM

```
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

Crédito: Autoria própria Utilizando o Arduino IDE

Por fim, criam-se variáveis para armazenamento dos dados de:

- Rede Wi-Fi e senha;
- Endereço IPv4 e IP, da rede Wi-Fi e do ESP32, respectivamente;
- Intervalo de tempo entre capturas de imagens e redefinição de tempo em relação à última imagem capturada.

Figura 48 - Programação da Captura de Imagens do ESP32-CAM

```
const char* ssid = "Lab 04 Arduino";
const char* password = "@phila2025@";
const char* serverNode = "http://192.168.0:28:8888/imageUpdate";
const char* serverESP  = "http://192.168.0.27/data";
const unsigned long captureInterval = 2000;
unsigned long lastCaptureTime = 0;
```

Crédito: Autoria própria Utilizando o Arduino IDE

void setup()

No “void setup”, iniciam-se a comunicação com o monitor serial e a conexão Wi-Fi entre o ESP32-CAM e a rede, representando esse processo no monitor serial já inicializado.

Figura 49 - Programação da Captura de Imagens do ESP32-CAM

```
Serial.begin(115200);
delay(1000);

Serial.println("Conectando ao WiFi...");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

Serial.println("\nWiFi conectado!");
Serial.println(WiFi.localIP());
```

Crédito: Autoria própria Utilizando o Arduino IDE

Inicia-se a câmera, configurando as funções de seus terminais.

Figura 50 - Programação da Captura de Imagens do ESP32-CAM

```
Serial.println("Inicializando câmera...");
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 10000000;
config.pixel_format = PIXFORMAT_JPEG;
```

Crédito: Autoria própria Utilizando o Arduino IDE

Analisa a PSRAM (Memória RAM) da câmera e continua o processo de inicialização com base na análise realizada, finalizando com um debug caso a câmera não inicialize.

Figura 51 - Programação da Captura de Imagens do ESP32-CAM

```
if(psramFound()){
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

if(esp_camera_init(&config) != ESP_OK){
  Serial.println("Falha ao inicializar a câmera!");
  while(true);
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

Cria a função de captura do ESP32-CAM.

Figura 52 - Programação da Captura de Imagens do ESP32-CAM

```
camera_fb_t* capture() {
  return esp_camera_fb_get();
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

void postingImage()

A biblioteca HTTPClient também é acionada, estabelecendo comunicação entre o ESP32-CAM e o servidor Node.js e definindo o tipo de conteúdo que será transmitido entre eles como ‘imagem jpeg”.

Figura 53 - Programação da Captura de Imagens do ESP32-CAM

```
void postingImage(camera_fb_t *fb){
  HTTPClient client;
  client.begin(serverNode);
  client.addHeader("Content-Type", "image/jpeg");
  int httpResponseCode = client.POST(fb->buf, fb->len);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Se o envio da imagem ao servidor for bem-sucedido, o JSON é retornado ao ESP32-CAM, senão a função é interrompida.

Figura 54 - Programação da Captura de Imagens do ESP32-CAM

```
if(httpResponseCode == 200){
  String response = client.getString();
  parsingResult(response);
} else {
  Serial.printf("Erro no servidor Node. Código HTTP: %d\n", httpResponseCode);
}
client.end();
```

Crédito: Autoria própria Utilizando o Arduino IDE

void parsingResult()

Essa função é responsável por verificar se a resposta em JSON é válida. Se for, os dados são utilizados, senão, o processamento é reiniciado.

Figura 55 - Programação da Captura de Imagens do ESP32-CAM

```
void parsingResult(String response){
  DynamicJsonDocument doc(4096);
  DeserializationError err = deserializeJson(doc, response);
  if (err) {
    Serial.println("Erro ao ler JSON do servidor!");
    return;
  }
```

Crédito: Autoria própria Utilizando o Arduino IDE

O servidor Node.js atribui rótulos à imagem capturada e retorna o resultado em JSON e exibe-o no monitor serial, como descrição e grau de confiabilidade.

Figura 56 - Programação da Captura de Imagens do ESP32-CAM

```
JsonArray array = doc.as<JsonArray>();
for (JsonVariant v : array) {
  JsonObject obj = v.as<JsonObject>();
  const char* description = obj["description"];
  float score = obj["score"];
  Serial.printf("%s: %.2f\n", description, score);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Quando o rótulo ‘flame’ (chamas) é identificado, o ESP32-CAM inicia um protocolo HTTP com o ESP32 e envia a informação para a segunda placa microcontroladora, adicionando um debug caso isso não seja feito.

Figura 57 - Programação da Captura de Imagens do ESP32-CAM

```
if (String(description) == "Flame" && score > 0.60) {
  HTTPClient http;
  http.begin(serverESP);
  http.addHeader("Content-Type", "application/json");
  String json = "{\"label\":\"FOGO\",\"score\":" + String(score, 2) + "}";
  int code = http.POST(json);

  if (code > 0)
    Serial.println("JSON enviado ao ESP32 receptor!");
  else
    Serial.printf("Falha ao enviar ao ESP32 receptor. Código: %d\n", code);
  http.end();
```

Crédito: Autoria própria Utilizando o Arduino IDE

void loop()

A variável ‘millis()’ (now) retorna o tempo em milissegundos a partir do início da compilação. Se o valor de millis() exceder o valor de ‘captureInterval’, então a variável ‘lastCaptureTime’ será redefinida e uma nova imagem será capturada.

Figura 58 - Programação da Captura de Imagens do ESP32-CAM

```
void loop() {
  unsigned long now = millis();
  if (now - lastCaptureTime >= captureInterval) {
    lastCaptureTime = now;

    camera_fb_t *fb = capture();
```

Crédito: Autoria própria Utilizando o Arduino IDE

Se a câmera não conseguir capturar mais imagens ou o formato não estiver em JPEG, a mensagem de erro é representada no monitor serial e o sistema é iniciado desde o começo.

Figura 59 - Programação da Captura de Imagens do ESP32-CAM

```
if (!fb || fb->format != PIXFORMAT_JPEG) {
  Serial.println("Falha ao capturar a imagem");
  esp_camera_fb_return(fb);
  return;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

A imagem é enviada ao servidor Node.js, e lá, processada e rotulada; e a memória do ESP32-CAM é esvaziado, recomeçando o processo.

Figura 60 - Programação da Captura de Imagens do ESP32-CAM

```
postingImage(fb);
esp_camera_fb_return(fb);
```

Crédito: Autoria própria Utilizando o Arduino IDE

A seguir, programação do envio das informações do ESP32-CAM para o ESP32 e posteriormente para o Telegram:

Circuito receptor (ESP32)

Pré-processamento:

Efetua-se a inclusão de bibliotecas:

- ESP32Servo (por Kevin Harrington)
- WiFi (por Arduino.cc)
- WiFiClientSecure (por Bc. Martin Chlebovec)
- ArduinoJson (por Benoit Blanchon)
- WebServer (por Khoi Hoang)

Figura 61 - Programação de Envio de Alerta Para o Telegram

```
#include <ESP32Servo.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include <WebServer.h>
```

Crédito: Autoria própria Utilizando o Arduino IDE

São adicionadas instâncias para as bibliotecas:

- ESP32Servo, como “servomotor” (para controlar o Servo-motor);
- WiFiClientSecure, como “client” (para a comunicação HTTP);
- WebServer, como “server(80)” (para criar o servidor HTTP).

Figura 62 - Programação de Envio de Alerta Para o Telegram

```
Servo servomotor;
WiFiClientSecure client;
WebServer server(80);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Variáveis para as credenciais da rede Wi-Fi, botToken e chatID do Telegram são criadas...

Figura 63 - Programação de Envio de Alerta Para o Telegram

```
const char* ssid = "EyeOnTree";
const char* password = "@phila2025@";
const String botToken = "8154918767:AAFcCBkpFNbGTm6JpZKEcDpVH6jMjsm4lBk";
const String chatID = "8040303412";
```

Crédito: Autoria própria Utilizando o Arduino IDE

... Em conjunto com algumas variáveis para a rotação do servo-motor e do alerta.

Figura 64 - Programação de Envio de Alerta Para o Telegram

```
bool alertActive = false;
int lastUpdateId = 0;
int angulo = 0;
bool subindo = true;
bool pararServo = false;
```

Crédito: Autoria própria Utilizando o Arduino IDE

void handleData()

Contém uma condição, onde será efetuada se o ESP32 receber uma informação do ESP32-CAM, representando-a no Monitor Serial como “alerta”.

Figura 65 - Programação de Envio de Alerta Para o Telegram

```
void handleData() {
  if (server.hasArg("plain")) {
    String body = server.arg("plain");
    Serial.println("ALERTA");
    Serial.println(body);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Além do “alerta” no Monitor Serial, o servo-motor é congelado, acendendo o LED-on-board do ESP32 e acionando a função de mensagem ao Telegram, explicada mais abaixo.

Figura 66 - Programação de Envio de Alerta Para o Telegram

```
  pararServo = true; // para o servo
  if (!alertActive) {
    digitalWrite(2, HIGH);
    sendTelegramMessageWithButton(
      "🔥 ALERTA! FOGO detectado por ESP32-CAM.\nClique abaixo para confirmar."
    );
```

Crédito: Autoria própria Utilizando o Arduino IDE

Um botão de “OK” é enviado juntamente com a mensagem do Telegram, para que o proprietário declare estar ciente do alerta. Então “waitForOk()” é a função de aguardo dessa confirmação, explicada mais abaixo.

Figura 67 - Programação de Envio de Alerta Para o Telegram

```
      alertActive = true;
      waitForOk();
    }
  }
  server.send(200, "text/plain", "OK");
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

void setup()

No setup, configurações iniciais são realizadas, como a configuração do pino de conexão do servo-motor (D5), e seu ângulo de direcionamento inicial.

Figura 68 - Programação de Envio de Alerta Para o Telegram

```
servomotor.attach(5);
servomotor.write(0);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Início da comunicação serial e configuração do pino do LED-on-board do ESP32 (D2) como saída e valor baixo.

Figura 69 - Programação de Envio de Alerta Para o Telegram

```
Serial.begin(115200);
pinMode(2, OUTPUT);
digitalWrite(2, LOW);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Conexão Wi-Fi do ESP32, colocando no Monitor Serial o status de conexão e o endereço IP do ESP32.

Figura 70 - Programação de Envio de Alerta Para o Telegram

```
WiFi.begin(ssid, password);
Serial.print("Conectando-se ao Wi-Fi");

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("\nWi-Fi conectado!");
client.setInsecure();

Serial.print("Endereço IP: ");
Serial.println(WiFi.localIP());
```

Crédito: Autoria própria Utilizando o Arduino IDE

Por fim, a inicialização do servidor HTTP, para comunicação entre o ESP32-CAM e o ESP32. O ESP32 é o servidor da comunicação, enquanto o ESP32-CAM é o cliente.

Figura 71 – Programação de Envio de Alerta Para o Telegram

```
  server.on("/data", HTTP_POST, handleData);
  server.begin();
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

void loop()

Aqui o código a ser repetido continuamente, onde “handleClient()” é configurado para monitorar as requisições e operações do servidor.

Figura 72 – Programação de Envio de Alerta Para o Telegram

```
void loop() {

  server.handleClient();
```

Crédito: Autoria própria Utilizando o Arduino IDE

A função if é utilizada para iniciar a rotação do servo-motor, para quando a variável “pararServo” não fosse verdadeira, realizando as próximas ordens.

Figura 73 – Programação de Envio de Alerta Para o Telegram

```
if (!pararServo) {
  servomotor.write(angulo);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Funções condicionais são utilizadas para a rotação do servo-motor, para quando a variável “subindo” fosse acionada.

Figura 74 – Programação de Envio de Alerta Para o Telegram

```
if (subindo) {
  angulo++;
  if (angulo >= 180) subindo = false;
} else {
  angulo--;
  if (angulo <= 0) subindo = true;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

O servo-motor rotaciona 1° a cada 50 milissegundos, tanto primeiramente de 0° a 180° e depois de 180° a 0°. Caso a variável “pararServo” for verdadeira, o servo-motor será desconectado do pino, congelando seu movimento.

Figura 75 – Programação de Envio de Alerta Para o Telegram

```
    delay(30);  // Move 1 grau a cada 50ms
  } else {
    servomotor.detach(); // Para o servo
  }

  delay(100);
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

void sendTelegramMessageWithButton()

Cria uma variável do tipo String para a URL do chat do Telegram, com a chave do bot.

Figura 76 – Programação de Envio de Alerta Para o Telegram

```
String payload = "{\"chat_id\":\"" + chatID + "\",";
payload += "\"text\":\"" + message + "\",";
payload += "\"reply_markup\":{\"inline_keyboard\":[[{\"text\":\"✅ OK\",\"callback_data\":\"/ok\"}]]}}";
```

Crédito: Autoria própria Utilizando o Arduino IDE

Figura 77 – Programação de Envio de Alerta Para o Telegram

```
String url = "https://api.telegram.org/bot" + botToken + "/sendMessage";
```

Crédito: Autoria própria Utilizando o Arduino IDE

Caso o ESP32 se conecte com o Telegram, a mensagem é enviada ao chat, utilizando HTTP.

Figura 78 – Programação de Envio de Alerta Para o Telegram

```
if (client.connect("api.telegram.org", 443)) {
  client.print(String("POST ") + url + " HTTP/1.1\r\n" +
               "Host: api.telegram.org\r\n" +
               "Content-Type: application/json\r\n" +
               "Content-Length: " + payload.length() + "\r\n" +
               "Connection: close\r\n\r\n" +
               payload);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Por fim, o status do envio da mensagem é representado no Monitor Serial, informando se foi enviado ou não (mensagem de erro). Caso não seja, o código possui um debug para continuar tentando novamente até que o envio seja concluído com êxito.

Figura 79 – Programação de Envio de Alerta Para o Telegram

```
    Serial.println("Mensagem enviada com botão inline ao Telegram.");
  } else {
    Serial.println("Erro ao conectar com o Telegram.");
  }
  client.stop();
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

waitForOk()

Função responsável pelo aguardo do sistema do ESP32 pela confirmação “OK” do usuário/proprietário.

Figura 80 – Programação de Envio de Alerta Para o Telegram

```
while (alertActive) {
  String url = "https://api.telegram.org/bot" + botToken + "/getUpdates?offset=" + String(lastUpdateId + 1);
```

Crédito: Autoria própria Utilizando o Arduino IDE

Se o ESP32 não conseguir se conectar ao Telegram, informará-o no Monitor Serial.

Figura 81 – Programação de Envio de Alerta Para o Telegram

```
if (!client.connect("api.telegram.org", 443)) {
  Serial.println("Falha ao conectar no Telegram.");
  delay(3000);
  continue;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

Valida também o servidor HTTP criado pelo ESP32.

Figura 82 – Programação de Envio de Alerta Para o Telegram

```
String payload;
while (client.connected()) {
  String line = client.readStringUntil('\n');
  if (line == "\r") break;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

Figura 83 – Programação de Envio de Alerta Para o Telegram

```
while (client.available()) {
  payload += client.readString();
}
client.stop();
```

Crédito: Autoria própria Utilizando o Arduino IDE

O JSON é utilizado na validação dos dados do ID e Token do chat do Telegram.

Figura 84 – Programação de Envio de Alerta Para o Telegram

```
DynamicJsonDocument doc(4096);
DeserializationError error = deserializeJson(doc, payload);
if (error) {
  Serial.println("Erro ao interpretar JSON.");
  delay(3000);
  continue;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

Figura 85 – Programação de Envio de Alerta Para o Telegram

```
JsonArray result = doc["result"];
for (JsonObject msg : result) {
  int update_id = msg["update_id"];
  lastUpdateId = update_id;
```

Crédito: Autoria própria Utilizando o Arduino IDE

Figura 86 – Programação de Envio de Alerta Para o Telegram

```
if (msg.containsKey("callback_query")) {
  String data = msg["callback_query"]["data"];
  String fromUser = msg["callback_query"]["from"]["first_name"];
```

Crédito: Autoria própria Utilizando o Arduino IDE

Se o botão de “OK” for clicado, o ESP32 será informado pelo Monitor Serial, apagando o LED-on-board e descongelando o movimento do servo-motor.

Figura 87 - Programação de Envio de Alerta Para o Telegram

```
Serial.print("Botão clicado: ");
Serial.println(data);

if (data == "/ok") {
  digitalWrite(2, LOW);
  sendTelegramMessage("✅ Alerta confirmado por " + fromUser + ". LED desligado.");
  alertActive = false;
  pararServo = false; // volta a girar o servo
  servomotor.attach(5);
  break;
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

void sendTelegramMessage()

Uma variável do tipo String é criada para o chat do Telegram, com o Token e ID, informados no pré-processamento.

Figura 88 - Programação de Envio de Alerta Para o Telegram

```
void sendTelegramMessage(String message) {
  String url = "https://api.telegram.org/bot" + botToken +
               "/sendMessage?chat_id=" + chatID +
               "&text=" + message;
```

Crédito: Autoria própria Utilizando o Arduino IDE

Se o ESP32 se conectar ao Telegram, o sistema vai requisitar a URL do servidor. A mensagem será enviada.

Figura 89 - Programação de Envio de Alerta Para o Telegram

```
if (client.connect("api.telegram.org", 443)) {
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: api.telegram.org\r\n" +
               "Connection: close\r\n\r\n");
```

Crédito: Autoria própria Utilizando o Arduino IDE

O status do envio da mensagem é representado no Monitor Serial. Caso não seja enviada a mensagem, tentará novamente até que seja concluído com êxito.

Figura 90 - Programação de Envio de Alerta Para o Telegram

```
    Serial.println("Mensagem enviada ao Telegram.");
  } else {
    Serial.println("Erro ao conectar com o Telegram.");
  }
  client.stop();
}
```

Crédito: Autoria própria Utilizando o Arduino IDE

7. Resultados e discussões

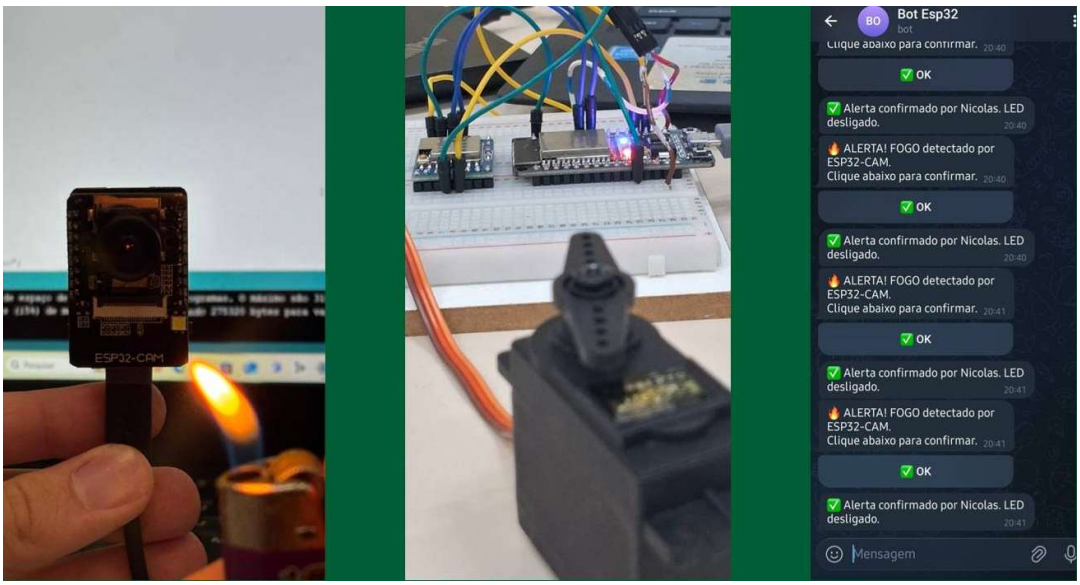
Foi elaborado um formulário para o público-alvo em relação à utilização de técnicas criativas para o combate de incêndios em áreas rurais e de cultivo. Recebeu-se respostas como o interesse de utilizar essas novas técnicas, bem como o compartilhamento de histórias relacionadas a perdas por incêndios.

Nos dias 29/10 e 30/10 ocorreu a Mostra Técnica, evento em que a escola apresenta os cursos oferecidos e os trabalhos realizados por professores e alunos, e o EyeOnTree fez parte dessa experiência. Escolas e famílias visitantes apresentaram interesse no projeto, em seu funcionamento e lógica, etc.

Figura 91 - Estande EyeOnTree Mostra Técnica 2025

Crédito: de autoria própria

Figura 92 - Imagens demonstrativas EyeOnTree



Crédito: de autoria própria utilizando o PowerPoint

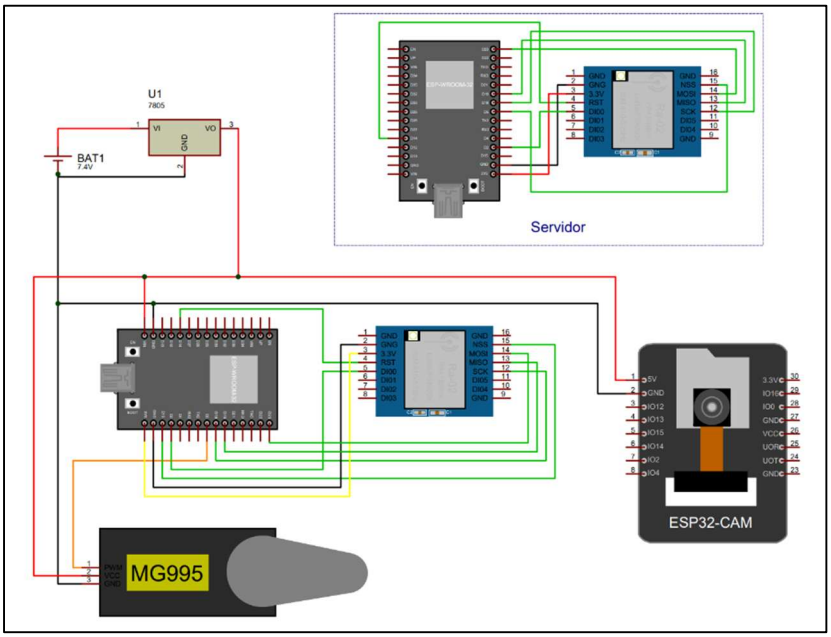
III. CONSIDERAÇÕES FINAIS

Ao final da montagem, o EyeOnTree se mostrou uma solução simples e funcional na detecção de incêndios, principalmente para pequenos produtores, que indispõe de técnicas avançadas e caras.

Notou-se o cumprimento com êxito dos objetivos para a proposta do trabalho, com exceção da implementação do LoRa, devido à complexidade da arquitetura e à priorização do funcionamento do protótipo com os atuais conhecimentos da equipe.

Algumas modificações podem ser realizadas para melhor aproveitamento e desempenho do projeto. A principal delas é a utilização de módulos LoRa, juntamente com outro ESP32, para alongamento da conexão, promovendo a comunicação entre sistemas a longa distância. Dessa forma, EyeOnTree se qualifica para a utilização em plantações de larga escala. A utilização de uma placa solar para recarregar a bateria periodicamente, dispensando a troca de baterias com frequência, e uma câmera infravermelha para a detecção dos incêndios por foco de calor, bem como a criação de uma inteligência artificial específica para o reconhecimento do fogo, são outras futuras melhorias que adequarão o projeto para real uso no mercado.

Figura 93 – Desenho do circuito EyeOnTree



Crédito: Autoria própria Utilizando o Proteus VSI

A imagem acima ilustra uma futura versão do projeto, utilizando os módulos LoRa e um ESP32 adicional.

EyeOnTree – IoT System for Monitoring Cellulose Forests Using ESP32-CAM and Telegram

Abstract:

The project in question consists of a fire monitoring system for pulp/silviculture forests. This system operates using the ESP32-CAM camera module with integrated artificial intelligence, in conjunction with an ESP32-based prototyping platform. Data transmission and reception is performed via Wi-Fi connection between the ESP32-CAM module, the ESP32, and the Telegram messaging application.

Keywords:

IoT; ESP32; ESP32-CAM; Forest Monitoring; Electronics; A.I.

REFERÊNCIAS BIBLIOGRÁFICAS

AVELAR, Julio. **Lora – Uma tecnologia que o software livre pode contribuir**. Disponível em https://embarcacoes.ic.unicamp.br/posts/lora/ . UNICAMP Embarcações, 27 abr. 2023. Acesso em 01 set 2025

BERTOLETI, Pedro. **O que é LoRa? Como funciona a rede e protocolo LoRaWAN**. MakerHero, 12 ago. 2024. Disponível em https://www.makerhero.com/blog/o-que-e-lora/ . Acesso em 01 set. 2025

CUNHA, João Pedro. **Proposta de um sistema de monitoramento de incêndios ambientais utilizando o ESP32-CAM.** Disponível em https://sol.sbc.org.br/index.php/eri-mt/article/view/18229/18063 **.** IFMT (Instituto Federal do Mato Grosso), 10 nov. 2021. Acesso em 23 fev. 2025.

DATATEM. **LoRa e LoRaWAN: conectividade de longo alcance para a Internet das Coisas**. https://datatem.com.br/lora-e-lorawan-para-iot/ . 11 abr. 2023. Acesso em 01 set. 2025

ESPBOARDS.dev. **AI Thinker ESP32-CAM Development Board – Guide and Datasheet**. Disponível em https://www.espboards.dev/esp32/esp32cam/ . Acesso em 26 ago. 2025

ESPRESSIF. **Espressif ESP32**. Disponível em: https://www.espressif.com/en/products/socs/esp32 .Acesso em 26 ago. 2025

ESPRESSIF. **ESP32 Series Datasheet**. Disponível em http://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf . Acesso em 26 ago. 2025

ESPRESSIF. **ESP32 Technical Reference Manual**. Disponível em http://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf . Acesso em 26 ago. 2025

MAIS FLORESTA. **Com maior oferta, preço do eucalipto e da celulose se estabiliza**. Mais Floresta, 06 jan. 2025. Disponível em https://www.maisfloresta.com.br/com-maior-oferta-preco-do-eucalipto-e-da-celulose-se-estabiliza/ . Acesso em 02 set. 2025

MARINO Store. **LM7805CV – Regulador de Tensão 5V.** Disponível em https://www.marinostore.com/componentes/lm7805cv-regulador-de-tensao-5v?srsltid=AfmBOoqULb_Z7zeD_1Js4o34DVhywLcmuleXPqHgPwefai7FuTikapJV . Acesso em 5 dez. 2025.

MOUSER Electronics. **Servo Motor Tower Pro MG995, 180 degrees**. Disponível em https://www.mouser.com/catalog/specsheets/Soldered_109042.pdf?srsltid=AfmBOooT_vo1xfRzavV3bOaca7Am7SSR-hVBOfgTszZYvOHLlO-iBaLK . Acesso em 01 set. 2025

PORTAL CELULOSE. **Incêndios devastam mais de 10 mil hectares de eucalipto em Água Clara (MS)**. Portal Celulose, 16 set. 2024. Disponível em https://portalcelulose.com.br/incendios-devastam-mais-de-10-mil-hectares-de-eucalipto-em-agua-clara-ms/ . Acesso em 02 set. 2025

SANTOS, Sara. **ESP32-CAM Save Picture in Firebase Storage.** Disponível em https://randomnerdtutorials.com/esp32-cam-save-picture-firebase-storage/ .Random Nerd Tutorials, 1 jul. 2025. Acesso em mar. 2025.

SHANKHDHAR, Priyansh. **DIY AI Camera with Google Vision & ESP32 CAM Module.** Disponível em https://how2electronics.com/diy-ai-camera-with-google-vision-esp32-cam-module/ . how2electronics, 29 mai. 2023. Acesso em

WIKIPÉDIA. **Centro Nacional de Prevenção e Combate aos Incêndios Florestais.** Disponível em https://pt.wikipedia.org/wiki/Centro_Nacional_de_Preven%C3%A7%C3%A3o_e_Combate_aos_Inc%C3%AAndios_Florestais?utm_source=chatgpt.com **.** 11 jun. 2025. Acesso em ago. 2025.