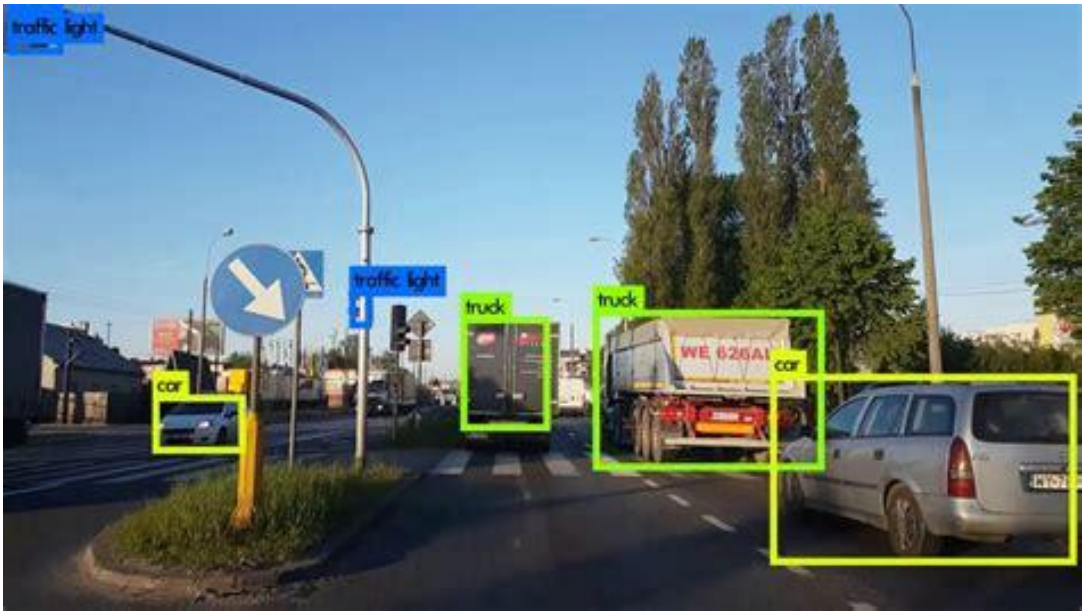
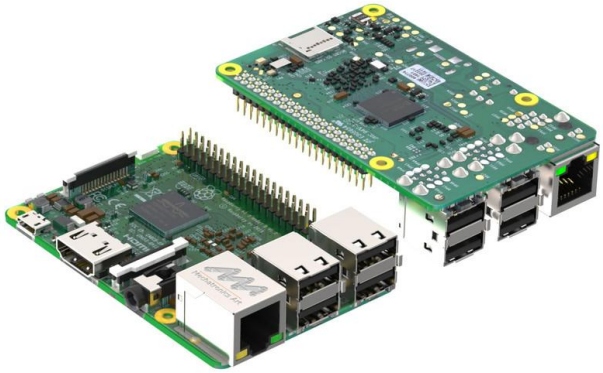
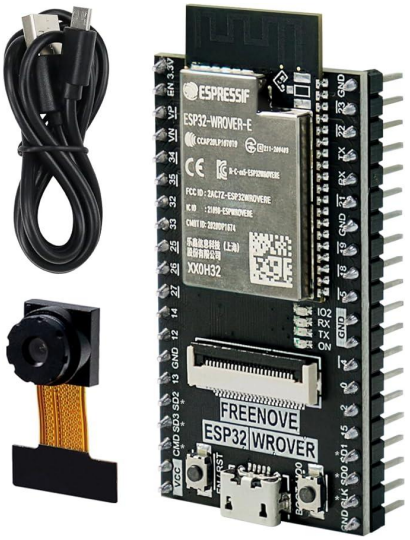
Relatório TP3 Sistemas Embebidos

Sistema de reconhecimento de pessoas e/ou objetos







Trabalho elaborado por:
Diogo Fernandes, Ivo Encarnação, Rafael Rio, Ricardo Fortuna
Nº22102558, Nº21901997, Nº22100521, Nº22107919

Introdução

Neste projeto, exploramos o uso do microcomputador Raspberry Pi e da placa de desenvolvimento ESP32 e certas comunicações entre si.

Com uma maior liberdade neste projeto, decidimos ir mais além e usar a câmara presente na placa ESP32, como “sensor”.

O objetivo principal foi criar um sistema que recolhe dados de imagem através da câmara, esses mesmos dados serão então enviados e processados no RaspberryPi, com finalidade de identificar os dados captados pela câmara.

Ao longo deste relatório, discutiremos a configuração utilizada no Raspberry Pi e na placa ESP32, e a lógica de programação implementada em ambos. No final deste documento, teremos uma visão abrangente do processo, bem como dos desafios enfrentados e soluções implementadas, proporcionando uma análise completa deste trabalho.

Descrição do projecto

A proposta feita pelo professor era desenvolver um projecto de tema aberto com os requisitos de usar um Raspberry Pi como núcleo do projecto, tirando proveito das suas características e funcionalidades.

Após uma pesquisa sobre o Raspberry Pi 3 B que foi o modelo escolhido para o projecto pelo grupo por um dos elementos possuir um, detectamos que as duas características que se destacava em relação ao arduino e o ESP32 era a posse de sistema operativo e um desempenho de processamento consideravelmente elevado comparando aos mesmos.

Foi então que surgiu a ideia de usarmos o processamento do raspberry pi para processar uma fotografia no sentido de começar uma investigação na área de Visão Computacional. Atualmente, essa área está presenciando um rápido crescimento em relação a tecnologia pela sua aplicabilidade, principalmente na segurança, no automatismo em casas inteligentes, cidades inteligentes, entre outras aplicações tecnológicas.

Para aumentar o nível de dificuldade do projeto, o grupo decidiu distribuir responsabilidades pelos aparelhos de maneira que usufrua do sistema operativo do Raspberry Pi e do seu processamento. Essa responsabilidade foi distribuída da seguinte maneira:

- Um computador faz uma ligação SSH para o Raspberry PI e corre o programa principal no Raspberry Pi.
- Raspberry Pi que tem um serviço de Servidor-Cliente com o ESP32 recebe a ligação SSH, corre o programa principal, e com o serviço de Servidor-Cliente pede os dados ao ESP32, assim que os dados são recebidos são analisados e a fotografia é guardada numa pasta específica.
- O ESP32 ficou responsável de tirar uma fotografia e enviar para o Raspberry Pi assim que essa seja solicitada, cumprindo assim o seu papel de Cliente.

A via de comunicação usada pelos 3 aparelhos foi o Wifi, os 3 aparelhos estavam conectados à mesma rede, caso contrário teria que ser usado uma VPN para resolver o problema de roteamento.

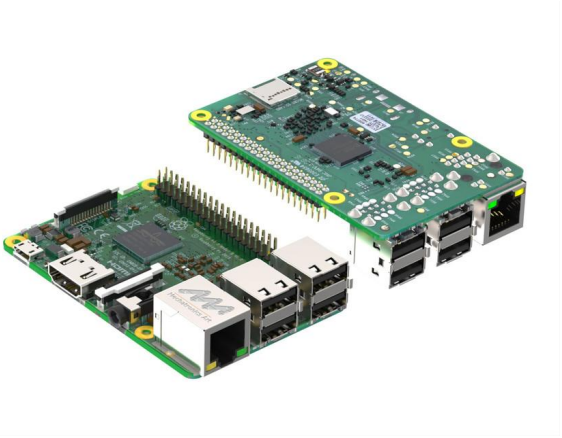
Componentes utilizados

Raspberry Pi

O Raspberry Pi 3 B é um microcomputador de placa única (SBC) desenvolvido pela Raspberry Pi Foundation. É uma placa de desenvolvimento popular para uma variedade de aplicações, incluindo educação, robótica, Internet das Coisas (IoT) e desenvolvimento de software.

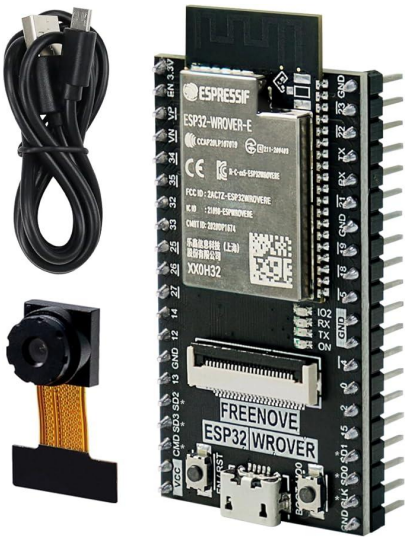
Características do Raspberry Pi 3 B:

- Processador quad-core de 64 bits, com uma frequência de 1,2 GHz.
- Wi-Fi e Bluetooth integrados.
- 1 GB de memória SDRAM.
- conectividade Ethernet 10/100
- Wi-Fi e Bluetooth integrados.
- Quatro portas USB 2.0
- Porta HDMI
- Possui 40 pinos GPIO (General Purpose Input/Output)



ESP32

O ESP32 Wrover é um microcontrolador de 32 bits desenvolvido pela Espressif Systems. É uma placa de desenvolvimento popular para a Internet das Coisas (IoT), oferece uma variedade de recursos que a tornam ideal para projetos de IoT. No entanto, o ESP32 tem 4MB de memória RAM que são divididos em 2 partes: uma para armazenar o código e outra parte para dados em tempo de execução.



Características do ESP32 Wrover:

- Processador dual-core de 32 bits com uma frequência de até 240 MHz.
- Wi-Fi e Bluetooth integrados.
- 4MB de memória RAM.
- GPIOs e periféricos

Camera_ESP32_OV2640

O ESP32-OV2640 possui uma câmera OV2640 de 2 megapixels que captura imagens de alta qualidade em resolução até 1600 x 1200. A câmera também possui um flash LED embutido para iluminação em ambientes escuros.

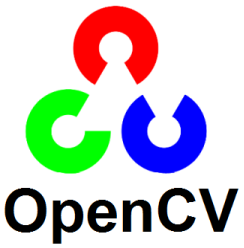
Tecnologias usadas

Python:

Python é uma linguagem de programação de alto nível, multiparadigma, suporta o paradigma orientado a objetos, imperativo, funcional e procedural. Possui tipagem dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens. O facto de ser uma linguagem dinâmica e muito acessível facilitou a implementação das outras tecnologias.

Bibliotecas

OpenCV:



OpenCV, originalmente, desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão Computacional.

YOLO(You Only Look Once):

O YOLO (You Only Look Once) é um algoritmo eficaz para detectar objetos em imagens e vídeos. Desenvolvido para funcionar em tempo real, adota uma abordagem de única passagem, dividindo a imagem numa grelha e fazendo previsões simultâneas de caixas delimitadoras e classes para cada célula da grelha. Esta metodologia proporciona rapidez e eficiência, sendo amplamente utilizada em aplicações como vigilância por vídeo e veículos autónomos. Com várias versões melhoradas ao longo do tempo, o YOLO destaca-se pela sua capacidade de detecção em tempo real e desempenho na visão computacional.

Código implementado

A implementação do código exigiu a divisão do mesmo por dois ficheiros distintos, evidenciando a complexidade da interconexão entre dispositivos, essa interconexão foi feita através de web server. O primeiro ficheiro foi destinado à plataforma Raspberry Pi, onde configuramos de forma meticulosa um serviço designado como "servidor". Neste contexto, o Raspberry Pi foi estrategicamente utilizado para desempenhar funções de alojamento e tratamento de dados, neste caso, no formato de imagem.

Por sua vez, o segundo ficheiro foi concebido para o ESP32, representando o componente designado como "cliente" no sistema. Nessa instância, o ESP32 foi configurado para desempenhar responsabilidades específicas em resposta às solicitações do servidor Raspberry Pi como tirar uma fotografia. Esta abordagem dual destaca a necessidade de uma arquitetura distribuída para a eficiente troca de dados entre os dispositivos.

Nome do ficheiro do Raspberry Pi: ‘Raspberry Pi Servidor’

Nome do ficheiro ESP32: ‘ ESP32 Cliente’

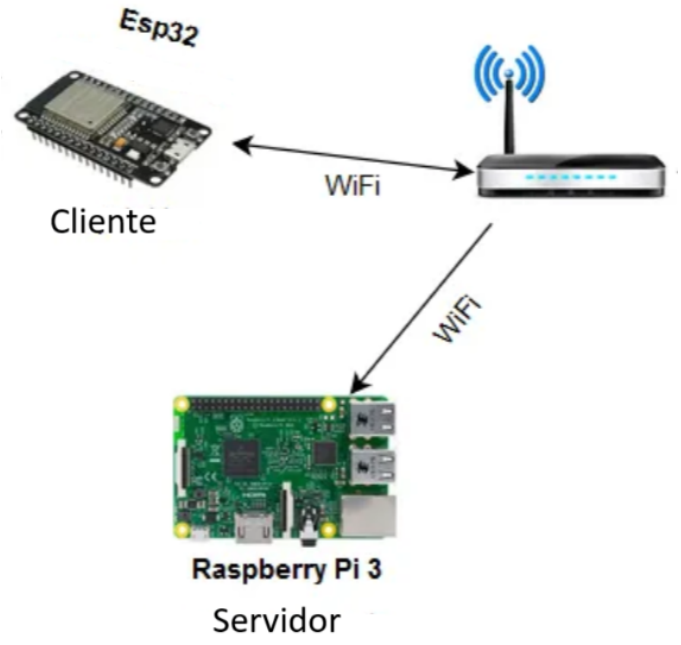
Diagrama esquemático/ esquema elétrico

O projecto é constituído por:

1 x Raspberry Pi

1 x Placa de ESP32 WROVER

1 x Câmera OV2640



Descrição do Código

ESP32:

Utilizamos as bibliotecas WebServer.h, WiFi.h e esp32cam.h. Definimos o SSID e a Password para a rede, definindo também que porta usamos.
O método serveJpg() será onde capturamos uma frame da câmara, caso não capte nada envia uma mensagem de erro, caso contrário ele envia uma mensagem de sucesso e grava a frame capturada. Os métodos handleJpgLo(), handleJpgHi() e handleJpgMid() apenas reconfiguram a captura para menos (Lo) ou mais (Hi) e média (Mid) resolução.

Setup():

Inicializa a comunicação serial para debug.

Configura a câmera com resolução de alta definição, definindo o buffer de imagens, qualidade JPEG e verifica se a câmera está operacional.

Configura a conexão Wi-Fi em modo estação (WIFI_STA) com as credenciais fornecidas (WIFI_SSID e WIFI_PASS).

Aguarda a conexão com a rede Wi-Fi ser estabelecida e, em seguida, imprime o endereço IP local da placa ESP32 e os caminhos para acessar as imagens JPEG geradas pela câmera.

Define handlers (funções) para atender às requisições HTTP nos caminhos especificados ("/cam-lo.jpg", "/cam-hi.jpg", "/cam-mid.jpg").

Inicia o servidor web.

Loop():

Continuamente chama a função server.handleClient() para atender às requisições de clientes web.

Raspberry Pi:

O código realiza a detecção de objetos na imagem especificada pela URL, desenha rectângulos delimitadores ao redor dos objetos encontrados, exibe o nome das classes e a confiança da detecção para cada objeto, e salva a imagem resultante com as marcações.

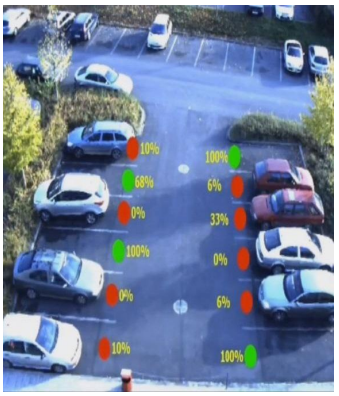
Utilizamos o YOLO (You Only Look Once) da biblioteca Ultralytics para detecção de objetos em imagens provenientes de uma URL específica.

Importação das bibliotecas necessárias: ultralytics.YOLO, cv2, math, urllib.request e numpy.
Definição da URL da imagem de onde os objetos serão detectados.

Carregamento do modelo YOLO pré-treinado utilizando o método YOLO() da Ultralytics, especificando o arquivo de pesos ("yolo-Weights/yolov8n.pt").
Criação da lista de nomes de classes para identificação dos objetos detectados.
Obtenção da imagem da URL especificada e decodificação dela num formato adequado para o OpenCV (cv2.imdecode).
Utilização do modelo YOLO para detectar objetos na imagem (results = model(im, stream=True)).
Iteração pelos resultados da detecção:
a. Extração das coordenadas das caixas delimitadoras (x1, y1, x2, y2).
b. Desenho das caixas delimitadoras na imagem original usando cv2.rectangle.
c. Exibição da confiança (probabilidade) da detecção e o nome da classe para cada objeto detectado.
d. Adição do texto na imagem usando cv2.putText.
Gravar a imagem resultante com os rectângulos delimitadores e rótulos dos objetos identificados (cv2.imwrite('test.jpg', im)).

Possíveis aplicações

O sistema que o grupo desenvolveu pode ser aplicado em vários contextos do dia a dia, tais como:



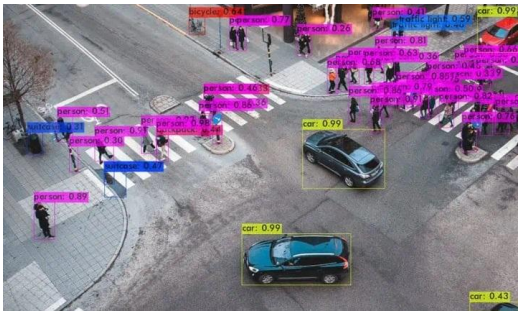
Segurança residencial ou empresarial, onde a tecnologia é utilizada para monitorizar e identificar atividades suspeitas, proporcionando alertas em tempo real para os proprietários ou responsáveis.

Parques de estacionamento inteligentes, onde o sistema pode identificar vagas disponíveis e informar os condutores em tempo real sobre a disponibilidade de estacionamento.

Monitorização de tráfego, utilizando câmeras para identificar congestionamentos, acidentes e violações de tráfego.

Análise de público em eventos, contando o número de pessoas e analisando padrões de movimento para otimizar a disposição do espaço e melhorar a segurança.

Gestão de resíduos, monitorando e otimizando a recolha com base na identificação automática de contentores cheios.



Problemas / Desafios

Este capítulo aborda os desafios e dificuldades enfrentados durante o desenvolvimento do projeto, destacando os obstáculos encontrados e as decisões tomadas para superá-los.

Comunicação entre Dispositivos

Um dos principais desafios enfrentados na fase inicial do projeto foi a definição do método de comunicação entre os dois dispositivos envolvidos. Inicialmente, o grupo escolheu usar sockets para estabelecer uma comunicação direta. No entanto, ao avançar na implementação, percebemos limitações e complexidades em arranjar informação perceptível associadas a esse método. A instabilidade na transferência de dados e a dificuldade de escalabilidade foram identificadas como pontos críticos. Diante disso, optamos pela abordagem baseada em web service, visando uma comunicação mais estável, segura e de fácil construção.

Investigação de Leitura de Imagem

Outro desafio importante foi na fase de pesquisa e desenvolvimento da capacidade de leitura de imagens. A escolha do algoritmo para processar eficientemente os dados visuais representou um obstáculo, dada a diversidade de opções disponíveis. A equipa enfrentou dificuldades na identificação de um algoritmo que proporcionasse um equilíbrio entre precisão e eficiência computacional. Este processo originou uma revisão da documentação, atrasando a implementação efetiva do módulo de visão computacional.

Algoritmo para Definir Objetos

A definição de um algoritmo robusto para identificar e classificar objetos também se mostrou desafiadora. A dificuldade desta tarefa, combinada à variedade de objetos a serem reconhecidos, obrigou a uma análise mais aprofundada destes algoritmos disponíveis. A busca pela precisão na identificação de objetos sem comprometer o desempenho geral do sistema exigiu um planeamento prévio da equipa pois, uma vez que o poder de processamento do Raspberry Pi 3b é limitado pelos seus componentes de CPU e memória SDRAM. Esta última foi também a razão pela qual decidimos gravar e analisar apenas um frame em vez de identificarmos objetos em “Live Streaming”.

Resultados/Discussão

Os desafios enfrentados durante o desenvolvimento forneceram valiosas lições aprendidas. A flexibilidade na escolha das tecnologias, a importância de uma pesquisa abrangente na fase inicial e a necessidade de uma abordagem iterativa na implementação de algoritmos foram aspectos destacados. As dificuldades enfrentadas foram desafios bastante enriquecedores não apenas o produto final, mas também a experiência técnica adquirida e o pensamento crítico da equipe ao enfrentar estes desafios.

Conclusão

Em conclusão, o desenvolvimento deste projeto envolveu uma série de desafios, desde a definição do método de comunicação entre dispositivos até a escolha de algoritmos para a identificação e classificação de objetos. Foi também bastante enriquecedor para o grupo, pois o projeto desafiou-nos a pesquisar novas bibliotecas, tanto na parte da comunicação entre dispositivos como no algoritmo de leitura e processamento de imagem. Contudo, os objetivos foram atingidos como esperado.

Este projeto, portanto, representa não apenas uma conquista técnica, mas também uma valiosa aprendizagem para futuras empreitadas tecnológicas.

WebGrafia / Bibliografia

https://www.udemy.com/course/raspberry-pi-for-beginners-step-by-step/
https://www.udemy.com/course/reconhecimento-de-textos-com-ocr-e-python/
https://www.udemy.com/course/deteccao-de-objetos-com-yolo-darknet-opencv-python/
https://towardsdatascience.com/evolution-of-yolo-yolo-version-1-afb8af302bd2