# Introdução

Esse semestre estou cursando as disciplinas de Programação Orientada a Objetos, Microcontroladores I e Redes de Computadores. As três irão realizar projetos finais práticos, e eu irei **integrar os três projetos finais em um só**. Cada disciplina será responsável por um componente independente do sistema, de forma que o sucesso em uma das disciplinas não depende das demais.

O projeto de Programação Orientada a Objetos será individual, o projeto de Redes de Computadores será realizado em dupla com o Rafael Nagel e o projeto de Microcontroladores I será individual.

# Overview do projeto

**Remote EXP**: exploração remota em ambientes hostis (exemplo: risco biológico)

Serão utilizados os carrinhos solares (figura abaixo) disponíveis no DAELN para desenvolver conceitos de inteligência artificial (navegação, exploração, visão computacional e deep learning), computação em nuvem (banco de dados, comunicação de dados, serviços web) e sistemas embarcados (microcontroladores, computação em edge, IoT).



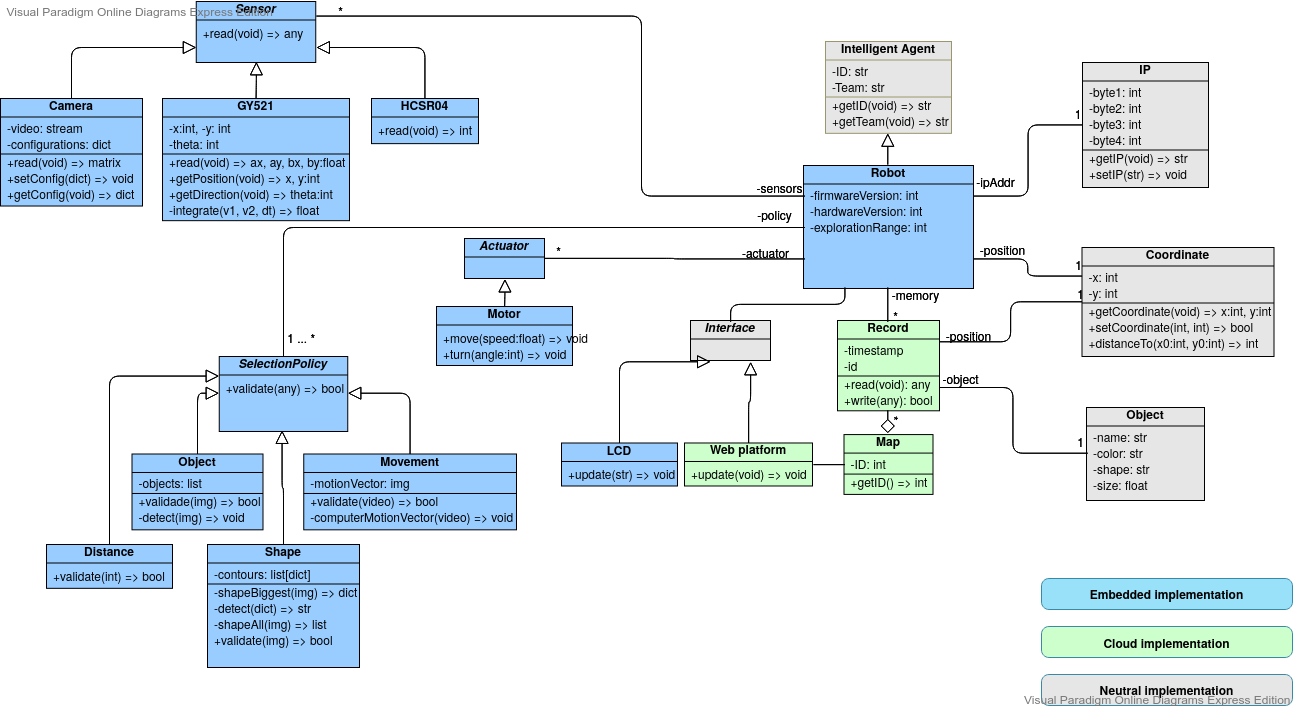
O carrinho andará em direções aleatórias, dentro de um raio máximo de exploração. O carrinho estará equipado com sensores de temperatura, umidade, pressão, acelerômetro, giroscópio, distância (na frente do carrinho) e **uma câmera de vídeo.**

Quando o carrinho se deparar com uma *situação de interesse*, irá parar. A discriminação entre situações interessantes e situações banais se dá por política de seleção, que escolhemos como sendo:

1. Há um objeto na frente do carrinho (distância<threshold); *ou*
2. A câmera detectou uma forma geométrica desenhada no chão; *ou*
3. A câmera detectou um objeto específico, como um rosto

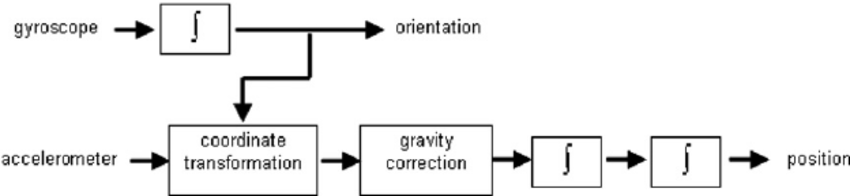
Quando uma situação de interesse é detectada, serão enviadas para o servidor a imagem da câmera, as coordenadas do carrinho e a leitura dos sensores ambientais (temperatura, umidade, pressão).

No servidor as imagens serão processadas usando Machine Learning para extrair mais informações das situações de interesse. Os registros (imagens, leituras ambientais e a posição do robô) ficarão armazenadas em um banco de dados e alguns resultado serão exibidos em uma página web. A figura abaixo mostra o diagrama UML do sistema, com as classes em azul estando relacionadas ao robô físico e as em verde relacionadas ao sistema web.



# Componente 1: navegação inercial

Serão utilizados acelerômetros e giroscópios para calcular a posição do carrinho enquanto ele explora o ambiente, utilizando o microcontrolador **MSP430** para ler e processar os sensores. O MSP estará dedicado unicamente a identificar a posição, visto que esta é uma operação delicada onde a frequência de integração influência muito da precisão da posição detectada. O processamento será implementado utilizando o filtro digital abaixo:



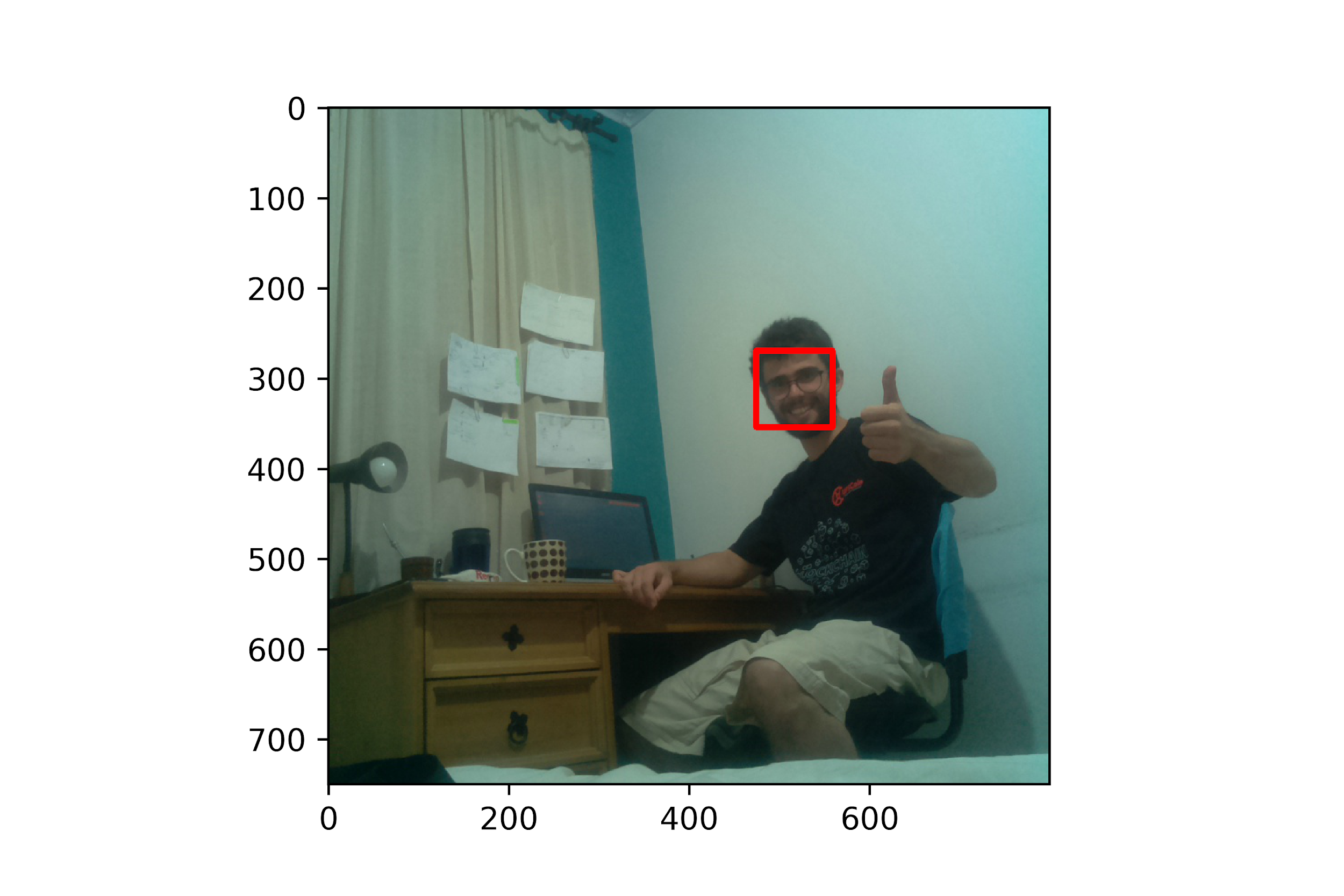
[Fonte](https://www.researchgate.net/publication/231092393_Pedestrian_dead_reckoning_employing_simultaneous_activity_recognition_cues)

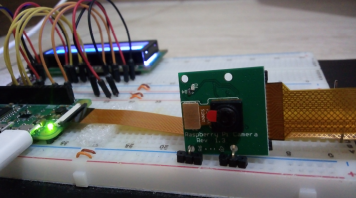
# Componente 2: Percepção e decisão

Um Raspberry pi está sendo utilizado para implementar o restante da lógica do robô, o que inclui:

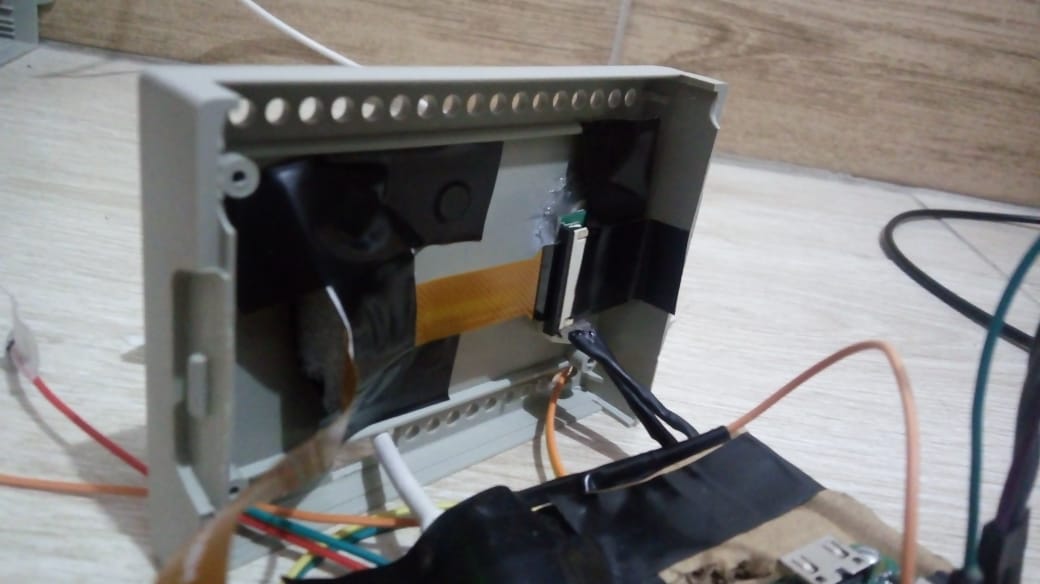
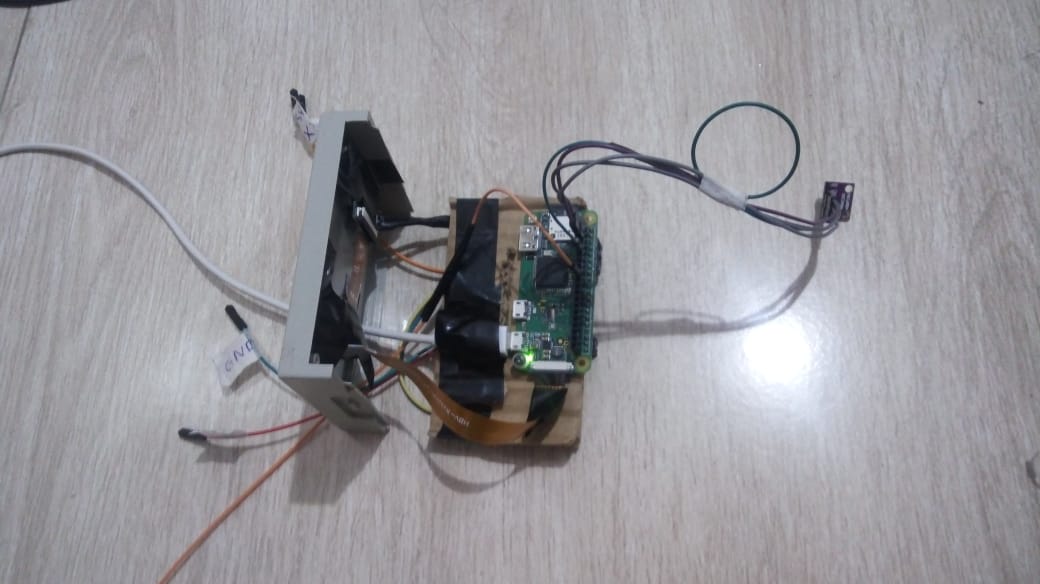
* Ler os demais sensores
* Controlar a movimentação do carrinho
* Identificar as situações de interesse

Atualmente temos implementado a leitura dos sensores, a identificação de formas geométrica (política de seleção 2), a identificação de rostos (o que poderia ser uma política de seleção, mas optamos por não utilizar-la), e a carcaça geral do código com a lógica de ler->selecionar->enviar.



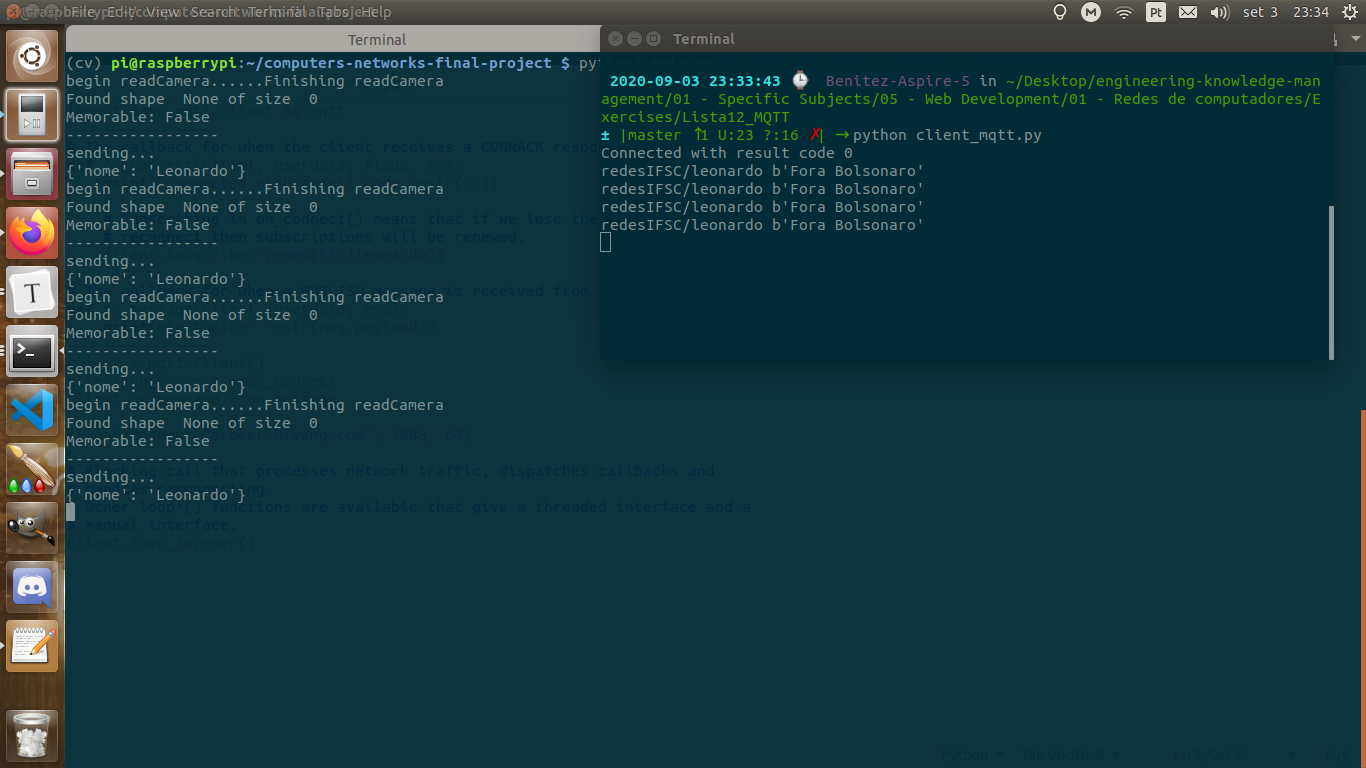


Após validação em matriz de contatos, colocamos em um gabinete:

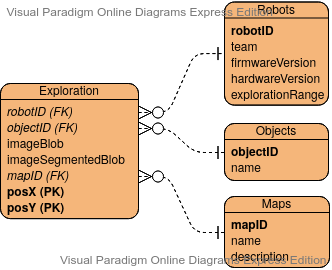


# Componente 3: Comunicação

É usado o protocolo MQTT para enviar os dados para um broker (hivemq):



Um servidor web está subscrito ao broker recebe a armazena os dados. Utilizou-se o framework Flask (python), conforme imagem abaixo de ele rodando localmente em docker. Um esboço inicial do diagrama Entidade-Relacionamento do banco de dados é mostrado abaixo.



# Componente 4: Análise posterior

As imagens recebidas (ou seja, apenas as selecionadas no edge) são processadas utilizando Redes Neurais Convolucionais para identificar quais objetos estão presentes na imagem e quais pixels da imagem pertencem a cada objeto (o que é chamado de **instance segmentation**), como mostra a figura abaixo.

