Reconhecimento Facial com Raspberry Pi

A ideia aqui é contar como foi implementado um sistema de reconhecimento facial, utilizando um Raspberry Pi 3, para controlar o acesso nos andares da CWI Software.

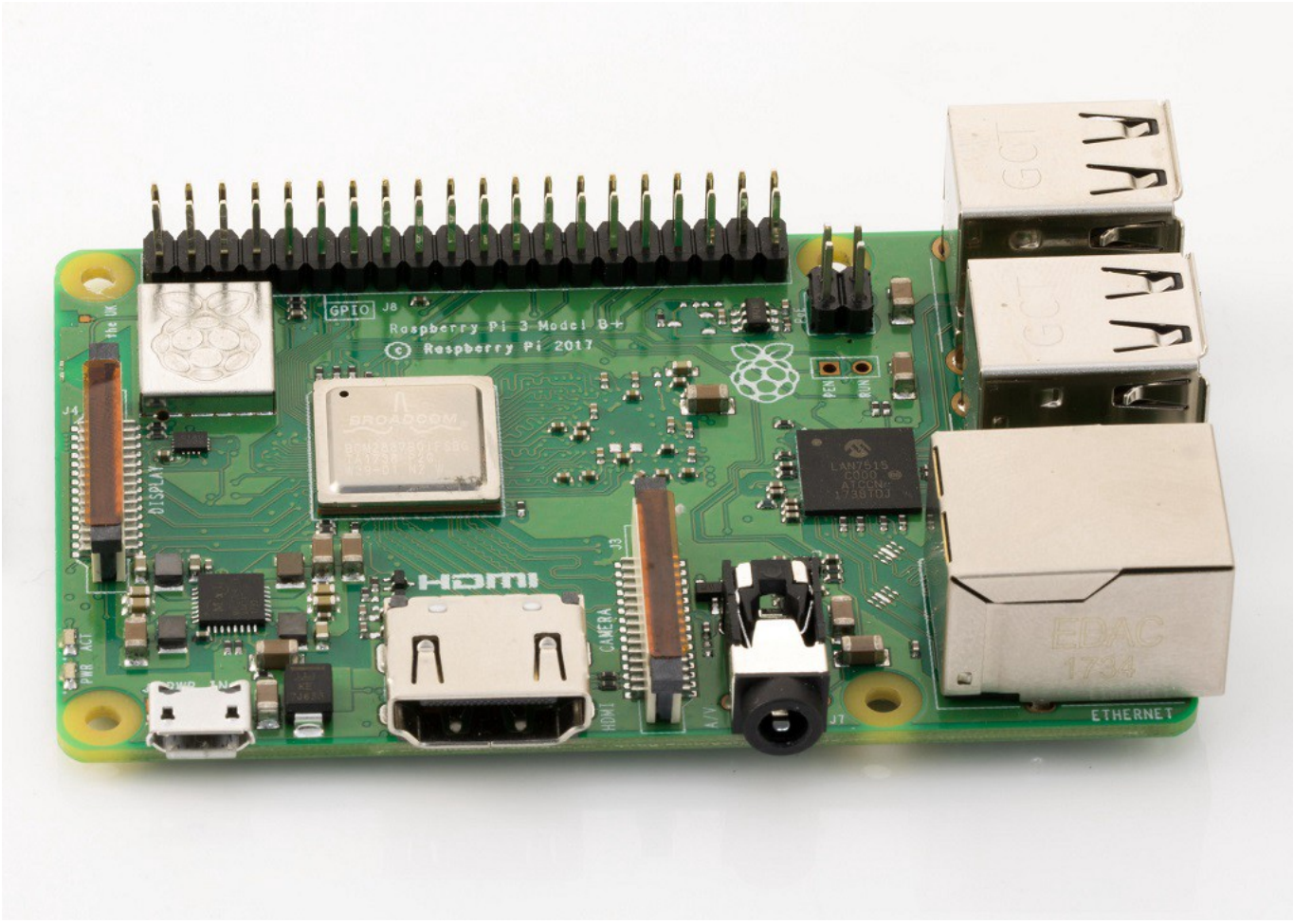
Para isso utilizamos um módulo de câmera específico para esta placa de desenvolvimento, que captura os frames de imagem. Foram usadas as bibliotecas *opencv* e *dlib* para buscar rostos nestes frames e mapear 64 pontos de referência, como olhos, nariz, boca e queixo. Para decidir se o rosto encontrado pertence a alguém cadastrado, é calculada a distância entre os pontos encontrados e os salvos no banco de dados. Esta distância deve estar abaixo de um valor fixo para definirmos com segurança que o rosto pertence à pessoa cadastrada.

Se você ainda não leu o artigo da Diandra falando sobre isso, corre lá e dá uma olhada, ela explica o problema e o processo de solucioná-lo é um pré-requisito para este artigo, já que vamos utilizar as mesmas tecnologias.

O Raspberry Pi

Antes de tudo, vamos ver o porquê do *Raspberry* ser uma alternativa para nosso projeto:

- É rápido, o modelo 3 b+ que estamos usando possui um processador quad-core de 1.4GHz e 1GB de memória RAM;
- É barato, custa menos de 300 reais e consome muito menos energia comparado a qualquer computador convencional;
- É capaz de acionar a porta diretamente com seu conjunto de *pinos GPIO* (vamos falar sobre isso logo mais);
- É pequeno e discreto;
- Possui um conector específico para o módulo de câmera, conectado diretamente na GPU, consumindo muito menos recursos da CPU se comparado a uma câmera USB tradicional;
- **E o mais importante**: É uma solução all-in-one, não vai depender da rede ethernet, ou qualquer serviço externo para reconhecer rostos e abrir portas.



O problema das soluções prontas

A Diandra explicou como funciona a lib *face_recognition*, que faz exatamente o que queremos fazer, então porque não utilizá-la aqui? Bem, há alguns pontos negativos a considerar: esta lib até funciona no Raspberry Pi, porém demora 10 segundos para processar uma imagem, encontrar um rosto e compará-lo com outros rostos.

Outro ponto negativo é que ela faz a comparação dos rostos em memória, então teríamos de carregar todos os rostos do banco de dados de uma vez só e consumir uma memória absurda neste processo, ou fazer uma consulta por vez ao banco, para cada rosto, gastando ainda mais tempo.

Pensando nisso decidimos usar diretamente as libs que a face_recognition utiliza por baixo dos panos: *opencv*, *dlib*.

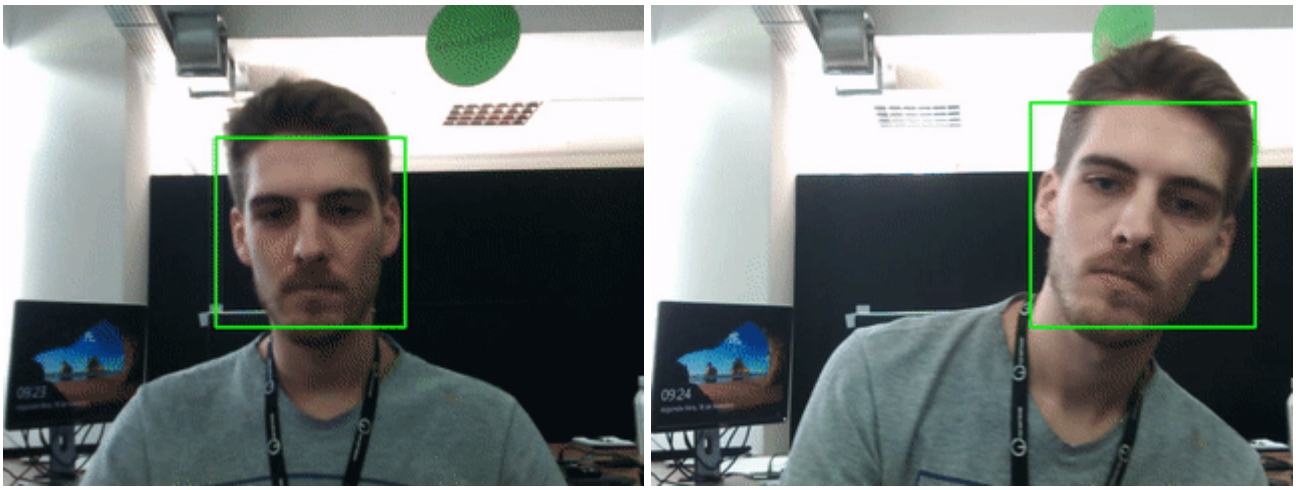
Planejando a solução

Primeiramente vamos analisar o *hardware*: uma **PiCamera**, o módulo de câmera oficial da Fundação Raspberry Pi, envia um stream de vídeo constante ao Raspberry Pi. Este faz todo o processamento necessário para detectar e comparar se um rosto pertence a um colaborador da CWI cadastrado; caso positivo, um relé é acionado através da GPIO e este relé abre a porta.

Todos os dispositivos estão conectados diretamente, deixando a solução independente da rede ethernet. O acionamento do relé pelo Raspberry é possível graças aos pinos GPIO (general-purpose input/output), com estes podemos ler ou definir níveis lógicos em tempo de execução, permitindo programas se comunicarem com o mundo externo.

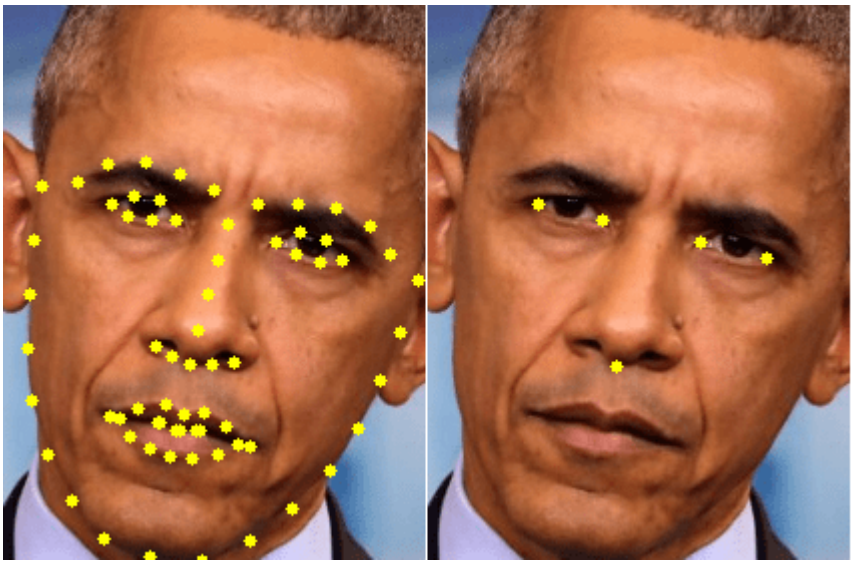
No Raspberry foi instalado o sistema operacional Raspbian, uma versão do Debian feita exclusivamente para funcionar nesta placa de desenvolvimento. O banco de dados utilizado foi o Postgres, já que permite o uso de arrays como tipo de variável, algo importante uma vez que cada rosto consiste em um array de 128 floats. Isto permite que cada linha de nossa tabela possua toda a informação necessária para comparar um rosto salvo com um detectado pela câmera. Ele está instalado localmente no Raspberry.

Para detectar rostos no frame da câmera utilizamos a opencv, invés da dlib usada na *face_recognition*. Podemos ver abaixo uma comparação de velocidade entre elas, mostrando que a opencv consegue ser 6 vezes mais rápida que a dlib.



Opencv (esquerda ~18 FPS) Dlib (direita ~3 FPS)

Com a posição do rosto podemos gerar os pontos de marcação essenciais para nossa comparação, existem dois modelos padrão: um que identifica 5 pontos e outro que leva em consideração 68 pontos.



Modelo de 64 pontos e 5 pontos faciais, respectivamente

Quantos mais pontos identificados por rosto, mais precisa será nossa comparação, por conta disso utilizamos o modelo de 68 pontos na solução com Raspberry Pi.

Usamos a *dlib* ainda para gerar os encodings do rosto, e são estes arrays únicos que utilizamos para comparar o rosto captado pela câmera com todos os rostos salvos. Esta comparação acontece no banco de dados, através de uma procedure que calcula a distância euclidiana entre todos os valores do array. O rosto que apresentar a menor distância é utilizado para comparação de semelhança, se esta distância for menor que um fator pré determinado é seguro afirmar que são a mesma pessoa.

Na situação de alguém ser identificado, um pino GPIO é setado para nível lógico alto que aciona o relé, e este a porta.

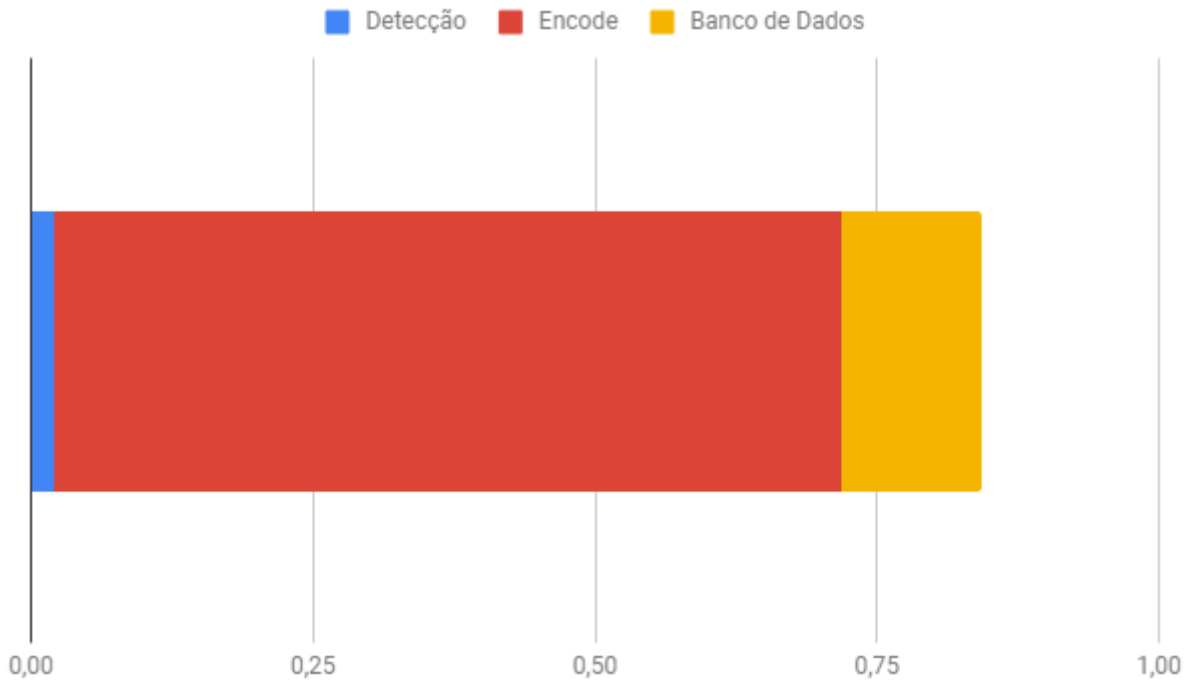
O cadastro de rostos é feito por um tablet; ele captura uma foto para inserir os encodings do rosto no banco. Optamos por realizar o merge desses dois bancos, onde importamos os rostos e usuários inseridos pelo tablet para o Raspberry Pi e exportamos o log de entradas. Isto para não sobrecarregar a placa com requisições de cadastro.

Resultados

Inicialmente instalamos o Raspberry Pi no 4° andar sem um monitor, agora ele está no 2º andar e adicionamos um monitor. No momento já temos 197 usuários cadastrados. O tempo médio para captar um frame de imagem, detectar um rosto nele, gerar 128 encodings e compará-los com todos os outros salvos no banco fica abaixo de 1 segundo (média de 0,85s).

O processo de buscar a imagem da câmera gasta um tempo praticamente irrisório se comparado aos seguintes (0,000114s), já o método da opencv que procura rostos nesta imagem consome um pouco mais de 0,02 segundos. Estas duas ações estão sempre sendo executadas e seu tempo de execução é quem define a quantidade de FPS do sistema.

Quando um rosto é detectado, são gastos 0,7 segundos para gerar seus encodings e mais 0,12s para compará-los no banco de dados.



Proporção de tempo gasto para cada tarefa

Durante o dia temos uma média de 330 acessos, com pico as 8:00 e cerca de 90 usuários distintos utilizando o sistema todo os dias:

Histograma com o número médio de entradas em um dia

Para utilizar o sistema basta cadastrar uma foto do seu rosto no tablet que está no Núcleo de Tecnologia.