# Lock-o-tron

**Rafael Sá Menezes[1], Rodrigo dos Santos Tavares[1], Felipe Leite Lobo[1], Herbert Rocha[1], Thais Oliveira Almeida[1], Leando N. Ballico[1]**

[1]Departamento de Ciência da Computação – Centro de Ciência e Tecnologia
Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brazil

{rafa.sa, rodrigo.tavares, felipe.lobo, herbert.rocha,

thais.oliveira}@ufrr.br,

leanball@gmail.com

***Abstract.*** *This paper describes the main features of Lock-o-tron, a security system based on facial recognition, while presenting its algorithms, modules, configuration, connections, libraries used and future works. The algorithms mentioned are Fisherfaces (for facial recognition) and fifo (for real-time scheduling). OpenCV was used as the facial recognition engine, and an Intel Galileo for controlling a motion sensor and an electric lock. At last, the system was able to achieve its goal.*

***Resumo.*** *Este artigo descreve as principais características do Lock-o-tron, um sistema de segurança com reconhecimento facial, abordando seus algoritmos, módulos, disposição, conexões, bibliotecas utilizadas e possibilidades de trabalhos futuros. Os algoritmos descritos são Fisherfaces (para reconhecimento facial) e o de fifo (para escalonamento em tempo-real). Foi utilizado OpenCV para fazer o reconhecimento dos rostos, e um Intel Galileo para controle de um sensor de presença e de uma fechadura elétrica. Ao final, concluímos que fomos capazes de atender a tarefa proposta.*

## Introdução

O Lock-o-tron é um projeto interdisciplinar que, unindo os conceitos de diversos campos da Computação, realiza o reconhecimento facial de pessoas. O sistema conta com: Um servidor, um Intel Galileo, um Yocto Linux configurado para *soft-realtime*, diversos sensores, diversos atuadores, uma interface Web para administração e um aplicativo Android para controle.

Este trabalho irá descrever as principais características do sistema, assim o leitor poderá entender melhor sobre o funcionamento do sistema como um todo.

## Dispositivos

O sistema é dividido em duas partes principais: O Galileo, responsável por controlar, e o servidor, responsável por administrar. A figura 1 contém o *Big Picture* do Lock-o-tron.

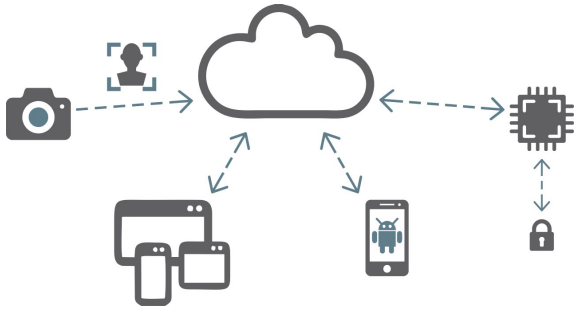

**Figura 1. Big Picture**

### Galileo

No Galileo conectamos:

**Sensor de Presença:** Responsável por detectar se existe alguma pessoa no local para que o sistema saiba quando deve iniciar o reconhecimento facial;

**Fechadura elétrica:** A fechadura é o atuador final do sistema, sendo ativada para abrir a porta.

A figura 2 mostra o esquema do circuito em um Galileo. Na aplicação real, porém, o LED seria substituído por uma fechadura elétrica, sem muitas alterações no circuito.

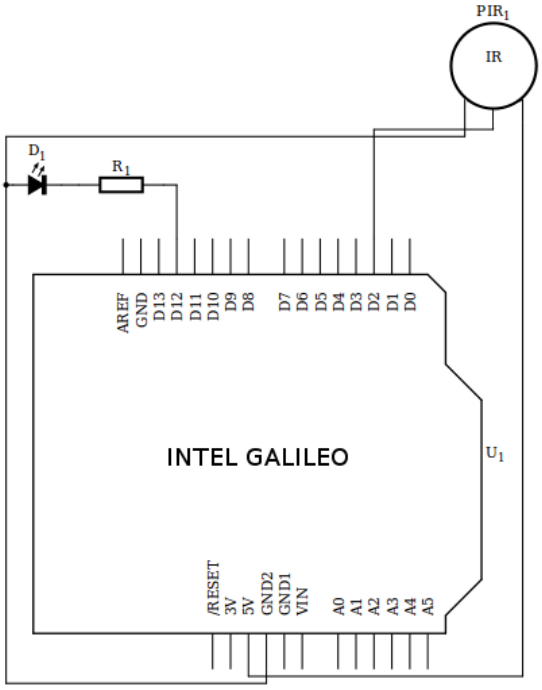


**Figura 2. Esquema do circuito do Galileo**

### Servidor

No Servidor conectamos:

**Câmera:** A câmera obtém os frames e os salva no servidor.

**Tabela 1. Orçamento**

| **Produto** | **Valor** |
|---|---|
| Intel Galileo | R$ 499,90 |
| Sensor de presença | R$ 18,90 |
| Câmera | R$ 200,00 |
| Fechadura elétrica | R$ 159,00 |
| Implementaçao do Sistema | R$ 6.120,00 |
| **Total:** | R$ 6.997,80 |

### Orçamento

O orçamento foi baseado em que cada programador receberia R$2000,00 mensais (sendo 20 dias úteis), trabalhando 6 horas por dia. O projeto levou em torno de 30 dias para ser desenvolvido. A tabela 1 mostra o orçamento do projeto

## Comunicação interna

Para comunicação do sistema foram utilizados: um servidor UDP, um *web service* e uma interface Web.

### Servidor UDP

Esse servidor UDP, que é executado no Galileo, é responsável pela interação de outros ambientes com o Galileo. Através destas mensagens UDP é possível abrir a porta e solicitar atualizações. A prioridade do servidor UDP é a mais alta no Galileo, então, sempre que alguém enviar uma mensagem, os outros processos serão interrompidos para que a ação pedida seja executada. A figura 3 mostra a rede de Petri que representa as computações do servidor UDP.

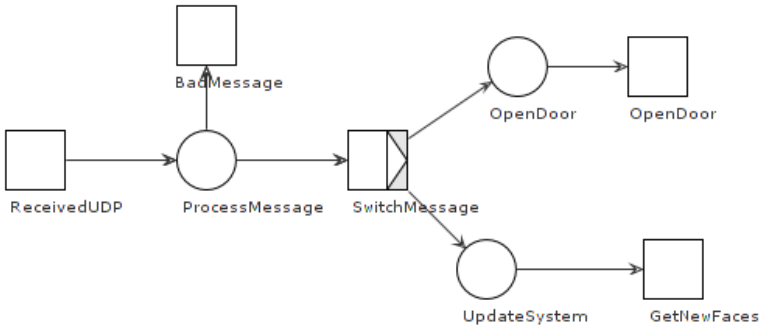


**Figura 3. Rede de Petri do servidor UDP**

### Web Service

O *web service*, executado no servidor, foi desenvolvido em PHP e é responsável pela interação de todos os ambientes entre si. As suas funções são:

- Enviar frames para o Galileo;
- Enviar atualizações para o Galileo;

- Enviar requisições UDP de abrir porta para o Galileo;
- Enviar requisições UDP de atualizar para o Galileo;
- Alterações no banco de dados (utilizando PDO);
- Requisições do banco de dados (utilizando PDO).

### Interface Web

A interface Web, executada no servidor, ficou responsável por ser um meio multiplataforma para modificar o banco de dados. Foi utilizado o *framework Material Design Lite* para CSS.

## Reconhecimento Facial

Para o reconhecimento facial, foram utilizados os algoritmos de *Fisherfaces* e *Eigenfaces*. Para 4 pessoas, obtendo 60 frames o *Fisherfaces* ocupou $534,5$KB's, já o *Eigenfaces* ocupou $31,4$MB's. Por se tratar de um sistema embarcado com espaço limitado, o algoritmo de *Fisherfaces* foi escolhido como definitivo. O Galileo obtém o frame do servidor (um frame já pré-processado) e atribui um ID a esse frame. Caso o ID seja 255, o frame é considerado como sendo de ninguém conhecido, caso contrário, é enviada uma requisição ao *web service* para descobrir se a pessoa está autorizada a entrar naquele horário. Caso ela esteja autorizada, a fechadura abre.

Os frames são pré-processados com as seguintes etapas:

1. Aplica-se uma conversão para tons de cinza;
2. Aplica-se uma equalização do histograma no frame;
3. Procura-se os dois olhos da imagem, continua apenas se encontrar;
4. Rotaciona-se o frame para os olhos ficarem alinhados no eixo x;
5. Aplica-se uma equalização do histograma nos lados direito e esquerdo do rosto;
6. Aplica-se o filtro de suavização;
7. Aplica-se uma máscara elíptica para remoção do pescoço.

Tudo isso para normalizar todos os frames e facilitar o reconhecimento facial.

O treinamento é feito ao pegar 60 frames pré-processados (da mesma pessoa) e associa-se essa pessoa a um ID (o mesmo utilizado no banco de dados), então é utilizado o algoritmo de *Fisherfaces* (que já vem implementado no opencv2) para o treinamento e é gerado um xml (treinado) com esses dados. Posteriormente, o Galileo obtém esse XML e aplica o algoritmo de *Fisherfaces* em um frame pré-processado. A figura 4 ilustra a rede de Petri das computações de reconhecimento facial no Galileo.

## Sistema Operacional

O sistema operacional utilizado no servidor é um GNU/Linux genérico, já no Galileo, é utilizado o Yocto Linux com o kernel configurado para *preempt*.

### Yocto Linux

O Yocto Linux é uma distribuição de Linux desenvolvida especificamente para sistemas embarcados. Para o Galileo, a Intel disponibiliza uma *recipe* específica, que dá suporte à placa e aos seus pinos.

Essa *recipe* adiciona uma biblioteca chamada *mraa*, que facilita a interação das portas, removendo a necessidade de se acessar as GPIO diretamente.

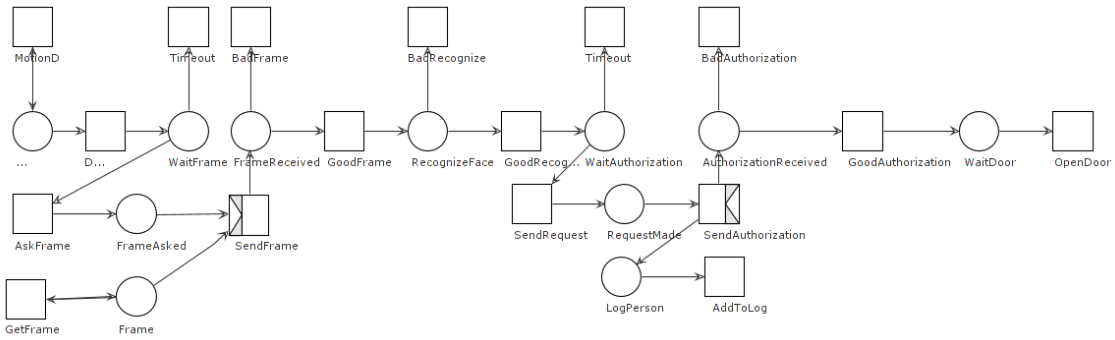


**Figura 4. Rede de Petri do Reconhecimento**

### PREEMPT - Tempo Real

O projeto utiliza um algoritmo de *soft-realtime*, já presente no Linux (configurado com *preempt*), o *fifo*. Este algoritmo tem prioridade sobre qualquer outra forma de escalonamento presente no sistema e executa o escalonamento da seguinte forma:

- Procura-se processo com maior prioridade FIFO (0 - menor, 100 - maior);
- Executa-se o processo até ele terminar ou receber uma mensagem de *yield*;

No Lock-o-tron, foi definido o servidor UDP com uma prioridade de 60 e o reconhecimento facial com prioridade 40. Todos os demais processos do sistema operacional não estão incluídos na tabela de tempo-real, então são os últimos a serem escalonados na tabela.

Por ser um algoritmo de tempo-real estático, o *fifo* não consegue garantir que irá cumprir todos os *deadlines*, mas, os resultados obtidos apenas com ele foram muito positivos.

## Aplicativo Android

O aplicativo utiliza o *web service* para comunicação com o Galileo, obtenção do histórico além de conter otimizações em baixo-nível para a arquitetura arm7. Conta com os seguintes módulos:

**Histórico:** Menu onde é possível visualizar os últimos usuários que tentaram entrar.
**Panic Button:** Botão de emergência, quando pressionado a porta abre;
**Update Button:** Botão para o Galileo atualizar seu arquivo de treinamento;
**Estatísticas:** Menu onde é possível ver diversas informações sobre o uso do sistema.

A otimização do aplicativo foi feita utlizando JNI e Java, com códigos em C e arm7, a parte otimizada foi a de cálculos estatísticos (presentes no aplicativo). A figura 5 mostra as telas do aplicativo e a figura 6 mostra o impacto das otimizações em baixo nível.

## Banco de Dados

O sistema utilizou um banco de dados SQL, o *MySQL*, para armazenamento dos dados. A figura 7 mostra o diagrama de entidade-relacionamento. Foi utilizado as seguintes tabelas:

**usuario:** Utilizada para armazenar o usuários do sistema;
**historico:** Utilizada para armazenar os *logs* de quem tentou entrar (armazenando se obteve acesso ou não) e qual a data/hora;
**autorizacao:** Utilizada para armazenar quais horários e dias os usuários estão autorizados a entrar.

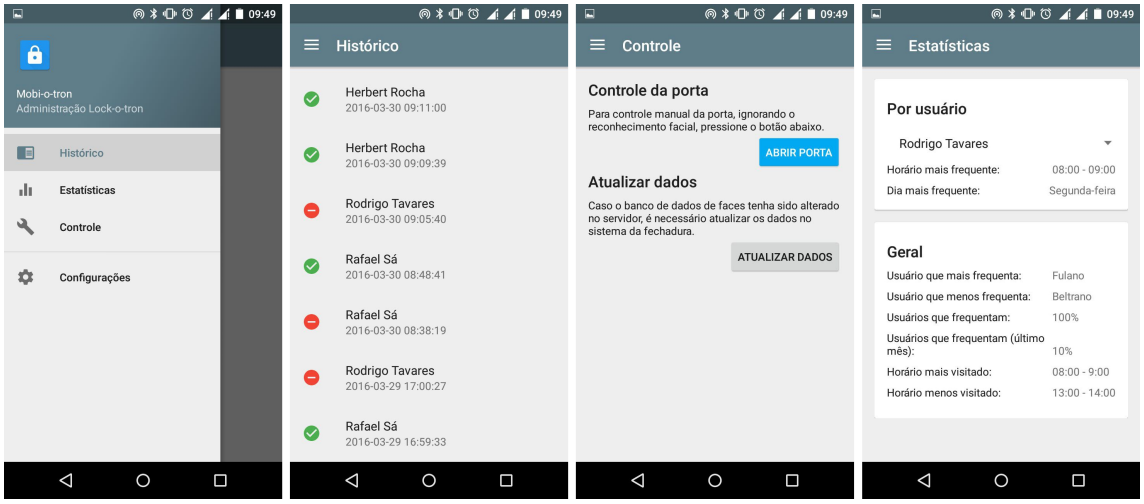


**Figura 5. Telas do aplicativo**

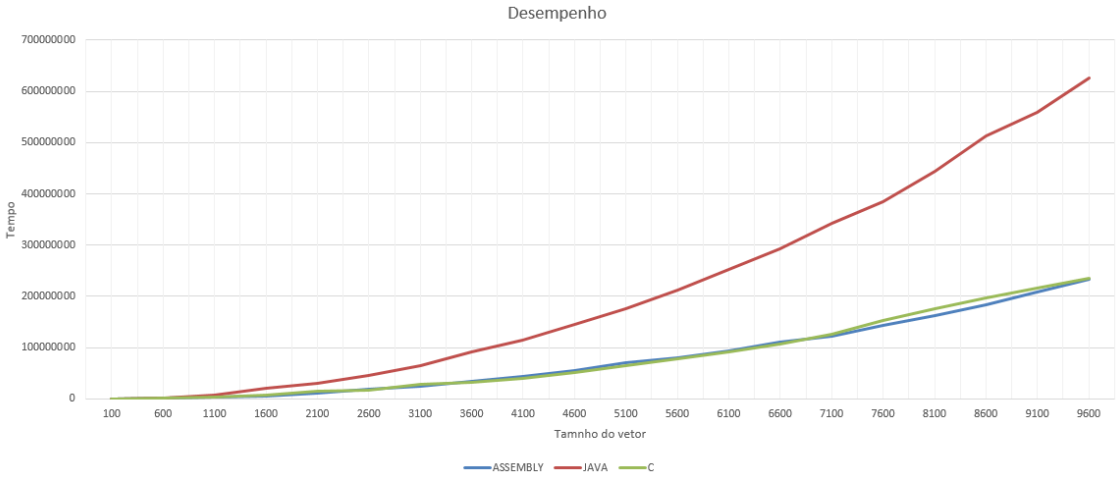


**Figura 6. Comparação de eficiência**

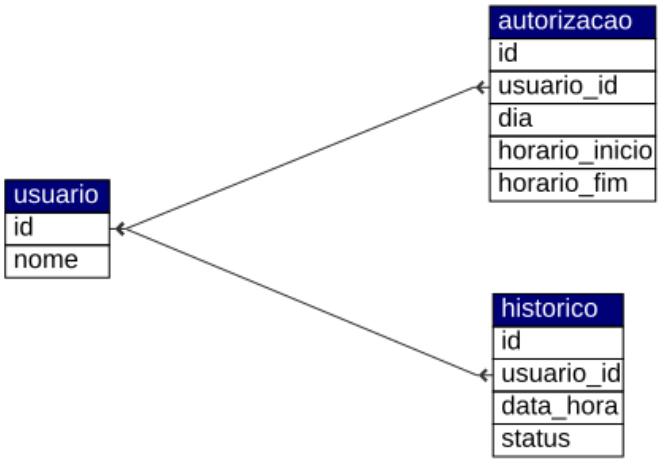


**Figura 7. Diagrama entidade-relacionamento**

## Trabalhos Futuros

O Lock-o-tron abre portas para diversas possibilidades de expansão:

- Melhorar o algoritmo de reconhecimento;
- Adicionar novos níveis de segurança;
- Verificar número de pessoas que estão entrando em conjunto;
- Automação da sala adaptável às preferências do usuário reconhecido;
- Um sistema para identificação de padrões de comportamento estranhos.

## Conclusão

Neste artigo foi apresentado as principais característica do Lock-o-tron, um sistema de segurança com reconhecimento facial, descrevendo seus algoritmos, módulos, disposição e bibliotecas utilizadas.

No final, o sistema criado é capaz de executar a tarefa proposta com certa precisão (ainda sendo ajustada), e o escalonador garante que por mais sobrecarregado o Galileo esteja, as tarefas sempre serão executados em tempo hábil.