

Projeto de Sistemas Embarcados

Sistema de controle de acesso de usuários integrado a nuvem

Angélica Kathariny de Oliveira Alves, Josué Bezerra Bonfim Filho

Resumo—De acordo com os requisitos necessários para o controle de acesso ao Laboratório de Engenharia e Inovação (LEI), este artigo tem como objetivo desenhar um produto cuja função principal é realizar o gerenciamento de acesso dos integrantes do LEI, com integração com e-mail e nuvem. O documento explicita o projeto de hardware e software do sistema, como é feita a coleta e o processamento de dados e o uso de API's de integração com sistemas externos. O sistema faz uso de uma Raspberry Pi Modelo B, um leitor de biometria e um display LCD 16x2.

Keywords—*Raspberry-pi, controle-de-acesso, biometria, API, e-mail*

I. INTRODUÇÃO

Um sistema de controle de acesso é capaz de gerenciar e controlar o fluxo de pessoas em áreas restritas. São amplamente utilizados em prédios residenciais e empresariais, academias, universidades e escolas.

Em muitos desses locais seu uso vai além de permitir ou não a entrada de um indivíduo em um determinado recinto. Em geral são utilizados para controle de frequência, cálculo de horas de trabalho, contagem precisa do número de usuários de um ambiente. Esses dados são importantes para uma boa administração do local onde o sistema foi implantado.

O sistema pode ser acionado por meio de cartões de acesso, autenticação por código numérico, leitura de código de barras ou reconhecimento biométrico. A maioria desses sistemas possui fácil instalação e operação do usuário porém possuem baixo nível de segurança, uma vez que as informações podem ser repassadas a terceiros sem autorização da equipe gestora do controle.

Com o intuito de elevar o nível de segurança dos sistemas o uso de reconhecimento biométrico vem sendo a melhor opção a ser implantada. Para isso a liberação de acesso é feita após a leitura e validação de características físicas particulares de cada indivíduo. Os métodos mais utilizados são leitura biométrica da mão, leitura da íris, leitura de impressão digital e reconhecimento facial.

O uso dessas tecnologias ainda propiciam integração com outros sistemas por intermédio softwares que facilitam a gestão e customização de acordo com as necessidades do gestor. Ademais as tecnologias de computação em nuvem e internet das coisas fazem com que essa gestão de informações seja feita remotamente e em tempo real, facilitando a tomada de decisões.

II. JUSTIFICATIVA

O Laboratório de Engenharia e Inovação - LEI representa um núcleo de laboratórios de pesquisa com a missão de produzir, desenvolver e difundir conhecimentos de Engenharias com responsabilidade social, transparência, inovação, ética e multidisciplinaridade. O objetivo geral do LEI é prover um espaço para a consolidação de pesquisa aplicada na área de Engenharias e inovação tecnológica no ambiente acadêmico.

Este núcleo é composto por seis laboratórios de pesquisa:

- Laboratório de Bioengenharia e Biomateriais - BioEngLab;
- Laboratório de Gerenciamento de Sistemas Dinâmicos;
- Laboratório de Computação Musical e Acústica;
- Laboratório de Estatística Aplicada à Probabilidade - LEAP;
- Laboratório de Instrumentação e Processamento de Imagens e Sinais - LIPIS;
- Laboratório de Informática e Saúde - LIS;

Integra, aproximadamente, 20 (vinte) pesquisadores e 100 (cem) alunos de graduação e pós-graduação que realizam pesquisa nas linhas de engenharia biomédica, biomateriais, ciência Mecânica, eletroeletrônica, engenharia de Software, informática em saúde, modelagem matemática e sistemas de controle. Possui diversos recursos tecnológicos como equipamentos de medição e análise, interfaces de interação humano-computador e bancadas de trabalho.

Para promover segurança aos usuários e ao patrimônio do LEI é necessário o uso de um sistema de controle de acesso. O mesmo já está implementado no LEI porém não possui muitos recursos de gerenciamento e não é integrado a uma rede de comunicação. Sua configuração permite apenas um administrador, cadastro de 150 usuários e não possui base de dados com informações dos usuários, apenas associa a impressão digital cadastrada a um número de identificação. Além disso, a solução tem apresentado problema em reconhecer digitais cadastradas, bloqueando o acesso de usuários.

III. OBJETIVO

O objetivo desse projeto é desenvolver um sistema de controle de acesso para o laboratório LEI que possua conectividade web. O produto trará diversos benefícios para a coordenação do LEI, como armazenamento de informação dos usuários, controle de horário de acesso e a inclusão e exclusão de novos membros no sistema. A integração com APIs de e-mail e nuvem trará facilidade também na comunicação com



Figura 1. Raspberry Pi 3 Model B

os usuários dos laboratórios.

A. Requisitos

O projeto possui os seguintes requisitos:

- Comunicação efetiva entre o leitor biométrico de impressão digital e a Raspberry-pi
- Integração entre a Raspberry-pi e a nuvem para o armazenamento das informações dos usuários
- Integração entre a Raspberry-pi e a API de email para a comunicação entre a coordenação do LEI e os usuários.

B. Descrição de Hardware

Este projeto fará uso dos seguintes componentes para a solução de hardware:

1) *Raspberry Pi 3 Model B*: A Raspberry Pi 3 Model B, como mostra a figura 1, possui custo médio aproximado de R\$ 200,00. Esta placa possui quatro portas USB, uma porta HDMI para conexão com display, um slot de cartão microSD para boot e armazenamento de dados, uma vez que a mesma não possui armazenamento interno.

Além disso, possui uma porta para conexão Ethernet com velocidades de 10/100 Mbit/s e WiFi para conexão com a internet. O sistema operacional usado é o Raspbian (Debian wheezy). RPi precisa de uma fonte alimentação com 5v e 3A(4.0 W) [7]. RPi 3 possui 1GB RAM and CPU com velocidade de 900 MHz quad-core ARM Cortex-A7.

2) *Leitor de Impressão Digital - FPM10A*: O projeto conta com um módulo de reconhecimento e gravação de impressão digital, como apresenta a figura 2. O módulo escolhido foi o FPM10A genérico fabricado na China. É alimentado por uma tensão DC que pode variar entre 3,3 e 6,0V, possui uma corrente de funcionamento menor que 120mA, e resiste a uma corrente de pico de até 140mA. O tempo para captura de imagens é de aproximadamente 1 segundo. Possui também uma tela para captura de imagem de 14x18mm. Este módulo tem capacidade de armazenamento de 1000 impressões digitais em sua memória. Seu microprocessador faz a busca das impressões cadastradas em 1 segundo e sua comunicação com outros dispositivos se dá via interface UART.



Figura 2. Leitor de Impressão Digital - FPM10A



Figura 3. LCD ITM-1602BSTL - INTECH LCD GROUP

3) *Display LCD16x2*: O display LCD é utilizado para informar o usuário quanto ao início e finalização dos processos de cadastro e busca de impressões digitais, informando também se o acesso é negado ou garantido. O modelo usado é o ITM-1602BSTL da INTECH LCD GROUP, ilustrado pela figura 5.

O modelo possui os pinos 1 e 2 para alimentação do LED BACKLIGHT+ e LED BACKLIGHT- respectivamente. Os pinos 3 e 4 são a alimentação do display. O pino 5 é usado para controlar o contraste do display e é ligado a um potenciômetro para esse ajuste. O pino 6 é o RS ou Register Select. O pino 7 é o Read/Write select e o mesmo deve ser aterrado. O pino 8 é o pino de Enable do Read/Write. Os pinos 9 a 16 são os DATA BUS. Para este projeto, a comunicação será feita por nibble, então apenas 4 pinos serão usados e estes são os pinos 13 a 16.

C. Descrição de Software

1) *Comunicação Raspberry Pi - PC com Linux*: Uma vez ligada, a RPi deve ser conectada a um computador via internet ou cabo Ethernet. Para a conexão via Ethernet, deve-se então dar o comando `$host raspberrypi` no terminal Linux para

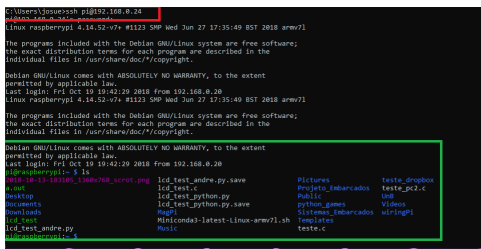


Figura 4. Teste da Conexão RPi - OS (Windows ou Linux)

descobrir o endereço ip da sua RPi. Após o comando aparecerá um aviso sobre a segurança da conexão e, se o usuário confia na conexão, deve portanto confirmar a mesma. Feito o login na RPi, tem-se total acesso ao dispositivo remotamente.

2) **Comunicação RaspberryPi - Display LCD:** Para esta seção, foi desenvolvido um script em Python para que a RPi envie ao LCD os comandos para que os caracteres sejam mostrados corretamente. A comunicação escolhida foi a do modo 4bits.

Esta etapa foi concluída facilmente através da biblioteca Adafruit_CharLCD. Esta biblioteca automaticamente seta os pinos de GPIO da RPi correspondentes à ligação dos pinos de dados RS, EN, D4, D3, D2 e D1. Também foi setado o número de colunas e linhas.

A biblioteca conta com várias funções para o gerenciamento do display, tais como as que limpam o display, seta cursor, habilita display, mostra o cursor, move para direita, move para esquerda, pisca cursor, escreve mensagem, entre outras.

3) **Comunicação RaspberryPi - DropBox:** Este pacote foi desenvolvido por Andrea Fabrizi. Trata-se de uma ferramenta *cross-plataform*, a qual suporta a API v2 do Dropbox. Não há necessidade de armazenamento de senhas de login. Ela possibilita que o usuário faça upload, download e delete de arquivos do Dropbox, além de outras funções.

Para a instalação, deve-se apenas clonar o repositório no GitHub, dar a permissão ao arquivo ".sh" e rodá-lo via terminal. Após esta etapa, deve-se então executar os arquivos com os parâmetros desejados.

4) **Comunicação Raspberry Pi - Sensor FPM10A:** A comunicação foi realizada, primeiramente, conectando a saída do sensor a um adaptador USB-TTL genérico e então conectado a porta USB da RPi. Após esta etapa, deve verificar se o dispositivo foi reconhecido pelas RPi. Em seguida deve-se instalar a biblioteca python-fingerprint e em seguida testar o sensor para verificar se tudo ocorreu como esperado.

IV. RESULTADOS E ANÁLISES

A. Comunicação RaspberryPi - Linux

A figura 4 mostra o prompt de comando do Windows 10 conectado com a RPi. Para que isso aconteça, precisamos encontrar o ip da RPi. Uma vez encontrado, e sabendo que o Windows 10 possui um cliente de ssh nativo, basta executar `> ssh pi@ip_da_RPi` aceitar a conexão e digitar a senha da RPi para que a conexão seja estabelecida.

Pode-se observar que o retângulo vermelho representa a etapa de conexão entre o Windows 10 e a RPi, enquanto o

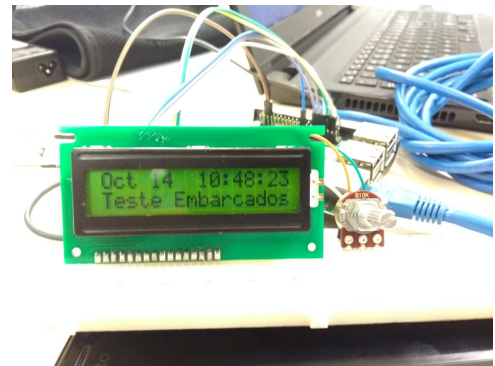


Figura 5. Teste da Conexão RPi - LCD 16x2

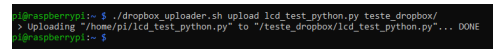


Figura 6. Teste da Conexão RPi - Dropbox no terminal

retângulo verde representa a etapa do controle total do terminal da RPi. Logo, pode-se concluir que a comunicação ocorreu da forma esperada.

B. Comunicação RaspberryPi - LCD 16x2

A figura 5 mostra o resultado do teste da comunicação RPi - LCD 16x2.

Através de um script em Python, foi possível enviar a data e a string "Teste Embarcados" para o display. Pode-se observar que a comunicação ocorreu da forma esperada.

C. Comunicação RPi - Dropbox

As figuras 6 e 7 mostram o resultado do teste da comunicação RPi - Dropbox.

Pode-se observar que a comunicação ocorreu de forma esperada.

D. Comunicação RPi - FPM10A

O teste foi realizado da seguinte forma: Um dedo não cadastrado no sistema foi colocado na tela do sensor. O mesmo leu a digital e informou que não há qualquer cadastro deste dedo. Então, o sistema cadastra o dedo na base de dados. O mesmo dedo que foi cadastrado foi recolocado na leitura do sistema, que retornou o número o qual a impressão digital foi cadastrada.

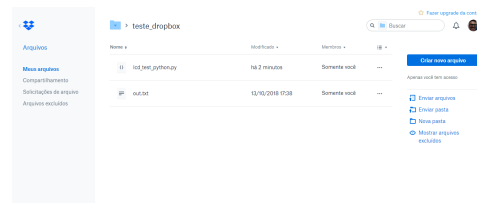


Figura 7. Teste da Conexão RPi - Dropbox no browser

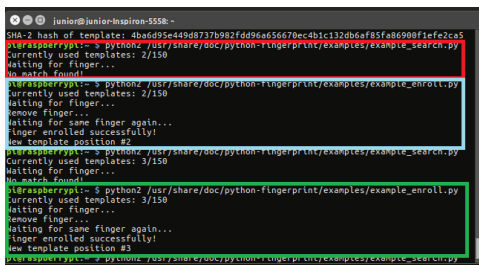


Figura 8. Teste da Conexão RPi - Sensor FPM10A

A figura 8 mostra o resultado deste teste, onde o retângulo vermelho indica a leitura do dedo não cadastrado, o retângulo azul indica o cadastro e o retângulo verde mostra que a digital foi cadastrada e foi reconhecida pela busca. A partir disso, afirma-se que o teste ocorreu de forma esperada.

V. CONSIDERAÇÕES FINAIS

Ao contemplar o material exposto neste documento, é possível notar que a base teórica para o projeto está consolidada e se mostra como um ponto de partida concreto para o desenvolvimento do sistema. Por meio de artigos, é possível notar que outras soluções com finalidades semelhantes a proposta nesse trabalho porém sem a comunicação necessária para o gerenciamento de usuários requerido pelo laboratório LEI. Com base no tempo e no escopo do projeto, conclui-se que o mesmo está dimensionado de acordo com o tempo de discussão proposto na disciplina de Sistemas Embarcados.

REFERÊNCIAS

- [1] FARIA, Diego Resende. Reconhecimento de impressões digitais com baixo custo computacional para um sistema de controle de acesso. 2005.
- [2] OLIVIA, Como funciona um sistema de controle de acesso?, 2015. Disponível em: <<http://www.graberalarmes.com.br/blog/como-funciona-um-sistema-de-controle-de-acesso/>>. Acesso em 04 de set. 2018.
- [3] Datasheet LCD16x2 INTECH ITM1602BSTL em: <http://www.intech-lcd.com/image/Character_LCM/1602b-c.pdf>. Acesso em 19 de out. 2018.
- [4] Datasheet Raspberry Pi 3 Model B. Disponível em: <<https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-B-plus-Product-Brief.pdf>>. Acesso em 19 de out. 2018.
- [5] Dropbox-Uploader. Disponível em: <<https://github.com/andreafabrizi/Dropbox-Uploader>>. Acesso em 19 de out. 2018.
- [6] Adafruit Python CharLCD. Disponível em: <https://github.com/adafruit/Adafruit_Python_CharLCD>. Acesso em 19 de out. 2018.
- [7] Python Fingerprint. Disponível em: <<https://github.com/bastianraschke/pyfingerprint>>. Acesso em 19 de out. 2018.

APÊNDICE

#Codigo do LCD

#!/usr/bin/python

from Adafruit_CharLCD import
↳ Adafruit_CharLCD

from time import sleep, strftime
from datetime import datetime
import socket

Initialize LCD (must specify pinout and
↳ dimensions)

lcd = Adafruit_CharLCD(rs=26, en=19,
d4=13, d5=6, d6=5, d7
↳ =11,
cols=16, lines=2)

def get_ip_address():

return [
(s.connect(('8.8.8.8', 53)),
s.getsockname()[0],
s.close()) for s in
[socket.socket(socket.AF_INET, socket.
↳ SOCK_DGRAM)]
][0][1]

try:

while 1:
lcd.clear()
ip = get_ip_address()
lcd.message(datetime.now().strftime(
↳ '%b_%d_%H:%M:%S\n'))
lcd.message('Teste_Embarcados')
sleep(2)

except KeyboardInterrupt:

print('CTRL-C_pressed. Program Exiting
↳ ...')

finally:

lcd.clear()

#Codigo para a procura de impresso digital

#!/usr/bin/env python
-*- coding: utf-8 -*-

"""

PyFingerprint

Copyright (C) 2015 Bastian Raschke <

↳ bastian.raschke@posteo.de>

All rights reserved.

"""

import hashlib

from pyfingerprint.pyfingerprint import
↳ PyFingerprint

Search for a finger

##

```

## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyUSB0',
        ↳ 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given
            ↳ fingerprint sensor password is
            ↳ wrong!')

except Exception as e:
    print('The fingerprint sensor could not
        ↳ be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f
    ↳ .getTemplateCount()) + '/' + str(f.
    ↳ getStorageCapacity()))

## Tries to search the finger and
    ↳ calculate hash
try:
    print('Waiting for finger...')

    ## Wait that finger is read
    while ( f.readImage() == False ):
        pass

    ## Converts read image to
        ↳ characteristics and stores it in
        ↳ charbuffer 1
    f.convertImage(0x01)

    ## Searches template
    result = f.searchTemplate()

    positionNumber = result[0]
    accuracyScore = result[1]

    if ( positionNumber == -1 ):
        print('No match found!')
        exit(0)
    else:
        print('Found template at position #'
            ↳ + str(positionNumber))
        print('The accuracy score is: ' +
            ↳ str(accuracyScore))

    ## OPTIONAL stuff
    ##

    ## Loads the found template to
        ↳ charbuffer 1
    f.loadTemplate(positionNumber, 0x01)

```

```

## Downloads the characteristics of
    ↳ template loaded in charbuffer 1
characterics = str(f.
    ↳ downloadCharacteristics(0x01)).
    ↳ encode('utf-8')

## Hashes characteristics of template
print('SHA-2 hash of template: ' +
    ↳ hashlib.sha256(characterics).
    ↳ hexdigest())

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)

#Cdigo para o Cadastro de nova Impresso
    ↳ Digital

#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
PyFingerprint
Copyright (C) 2015 Bastian Raschke <
    ↳ bastian.raschke@posteo.de>
All rights reserved.

"""

import time
from pyfingerprint.pyfingerprint import
    ↳ PyFingerprint

## Enrolls new finger
##

## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyUSB0',
        ↳ 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given
            ↳ fingerprint sensor password is
            ↳ wrong!')

except Exception as e:
    print('The fingerprint sensor could not
        ↳ be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f
    ↳ .getTemplateCount()) + '/' + str(f.
    ↳ getStorageCapacity()))

```

```

## Tries to enroll new finger
try:
    print('Waiting for finger...')

    ## Wait that finger is read
    while ( f.readImage() == False ):
        pass

    ## Converts read image to
    ↪ characteristics and stores it in
    ↪ charbuffer 1
    f.convertImage(0x01)

    ## Checks if finger is already enrolled
    result = f.searchTemplate()
    positionNumber = result[0]

    if ( positionNumber >= 0 ):
        print('Template already exists at
            ↪ position #' + str(
            ↪ positionNumber))
        exit(0)

    print('Remove finger...')
    time.sleep(2)

    print('Waiting for same finger again
        ↪ ...')

    ## Wait that finger is read again
    while ( f.readImage() == False ):
        pass

    ## Converts read image to
    ↪ characteristics and stores it in
    ↪ charbuffer 2
    f.convertImage(0x02)

    ## Compares the charbuffers
    if ( f.compareCharacteristics() == 0 ):
        raise Exception('Fingers do not
            ↪ match')

    ## Creates a template
    f.createTemplate()

    ## Saves template at new position
    ↪ number
    positionNumber = f.storeTemplate()
    print('Finger enrolled successfully!')
    print('New template position #' + str(
        ↪ positionNumber))

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)

```