UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Faculdade de Engenharia Mecânica

Graduação em Engenharia Mecatrônica

Sistemas digitais para Mecatrônica

Prof.: Éder Alves de Moura

Trabalho 2

Baltazar Alic Borges da Silva – 11711EMT022

Rodrigo Alves Prado – 11521EMT003

Fernando Rabelo Fernandes Junior – 11611EMT020

2022

1. FLUXOGRAMA

2. FUNÇÕES

A aplicação tem como controle um site hospedado no heroku, onde existem 4 botões que quando pressionados trocam os estados das lâmpadas de um cômodo da casa. O site pode ser visto na imagem abaixo:

O servidor rodando no Linux tem o papel de colher os estados das lâmpadas e passá-los para os clientes criados pelo programa que gera a interface gráfica e assim simular o controle das lâmpadas de uma casa.

A interface gráfica é mostrada abaixo:

Os nomes dos cômodos abaixo das imagens também são botões que ao serem pressionados mudam o estado da lâmpada do cômodo e altera o banco de dados no heroku.

3. REFERÊNCIAS

https://flask.palletsprojects.com/en/2.0.x/

https://www.youtube.com/watch?v=Z1RJmh_OqeA

https://www.youtube.com/watch?v=Z1RJmh_OqeA

https://www.youtube.com/watch?v=VhhNlWdLPzA

https://www.youtube.com/watch?v=Su4nRkiW2NQ&list=PLYj9OBozaypbr9jRLGWmCAODbX1Vr5bb6

4. APÊNDICE

Código HTML:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" content="width=device-width, initial-scale=1.0">
   <title>controle</title>
   <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='estilo.css') }}">
</head>

<body bgcolor="#000000" class="body">
   <h1>
      <img src="{{ url_for('static', filename='lightboost.png') }}"
width="358" height="95" alt=""/>
   </h1>
   <h2 align="center">
      <img src="{{ url_for('static', filename='texto1.png') }}" width="543"
height="48" alt=""/>
   </h2>
   <table border="0" align="center">
     <tbody>
      <tr>
        <th scope="col" height="100"> </th>
      </tr>
      <tr>
        <th scope="row">
           <form method="post" action="/comodos/sala">
           <input type="submit" class="botao" id="sala_submit">
           <center class="img"><label for="sala_submit" class="label">
              {% if sala == "True" %}
```

```
              <img src="{{ url_for('static', filename='sala_lig.png') }}"
width="350" height="330" alt=""/>
              {% else %}
               <img src="{{ url_for('static', filename='sala.png') }}"
width="350" height="330" alt=""/>
              {% endif %}
           </label></center>
           </form>
        </th>
        <td align="center">
           <form method="post" action="/comodos/cozinha">
           <input type="submit" class="botao" id="cozinha_submit">
           <center class="img"><label for="cozinha_submit" class="label">
              {% if cozinha == "True" %}
               <img src="{{ url_for('static', filename='cozinha_lig.png') }}"
width="350" height="330" alt=""/>
              {% else %}
               <img src="{{ url_for('static', filename='cozinha.png') }}"
width="350" height="330" alt=""/>
              {% endif %}
           </label></center>
           </form>
        </td>
      </tr>
      <tr>
        <th scope="col" height="100"> </th>
      </tr>
      <tr>
        <th scope="row">
           <form method="post" action="/comodos/quarto">
           <input type="submit" class="botao" id="quarto_submit">
           <center class="img"><label for="quarto_submit" class="label">
              {% if quarto == "True" %}
               <img src="{{ url_for('static', filename='quarto_lig.png') }}"
width="350" height="330" alt=""/>
              {% else %}
               <img src="{{ url_for('static', filename='quarto.png') }}"
width="350" height="330" alt=""/>
              {% endif %}
           </label></center>
           </form>
        </th>
        <td align="center">
           <form method="post" action="/comodos/banheiro">
           <input type="submit" class="botao" id="banheiro_submit">
           <center class="img"><label for="banheiro_submit" class="label">
              {% if banheiro == "True" %}
               <img src="{{ url_for('static', filename='banheiro_lig.png') }}"
width="350" height="330" alt=""/>
              {% else %}
               <img src="{{ url_for('static', filename='banheiro.png') }}"
width="350" height="330" alt=""/>
              {% endif %}
           </label></center>
           </form>
        </td>
      </tr>
     </tbody>
```

```
   </table>
</body>
</html>
```

Código flask:

```
from flask import Flask, render_template, request, redirect
from flask_sqlalchemy import SQLAlchemy
import threading
import time

DB_NAME = "lamps.db"
app = Flask(__name__, template_folder='template')
app.config['SECRET_KEY'] = 'hjshjhdjahkjshkjdhjs'
app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_NAME}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
db.init_app(app)

#classe que representa uma tabela do banco de dados
class Devices(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(30), nullable=False)
    state = db.Column(db.String(10), nullable=False)

    def __init__(self, id, name, state):
        self.id = id
        self.name = name
        self.state = state

"""
db.create_all()

sala = Devices(1, "sala", "False")
cozinha = Devices(2, "cozinha", "False")
quarto = Devices(3, "quarto", "False")
banheiro = Devices(4, "banheiro", "False")

db.session.add(sala)
db.session.add(cozinha)
db.session.add(quarto)
db.session.add(banheiro)


db.session.commit()
"""
#tuplas do banco de dados que representam as lampadas
sala = Devices.query.filter_by(id=1).first()
cozinha = Devices.query.filter_by(id=2).first()
quarto = Devices.query.filter_by(id=3).first()
banheiro = Devices.query.filter_by(id=4).first()

#pagina raiz do site
@app.route("/")
def control_panel_route():
    global app, db
```

```
    state = []
    db.init_app(app)
    #capitura os status das luzes gravadas no banco de dados
    for i in range(4):
        var = Devices.query.filter_by(id=i+1).first()
        state.append(var.state)
    return render_template('control.html',
                           sala=state[0],
                           cozinha=state[1],
                           quarto=state[2],
                           banheiro=state[3])

#pagina para alterar o banco de dados quando um botão é pressionado
@app.route("/comodos/<string>", methods=["POST", "GET"])
def sala_edit(string):
    global app, db
    db.init_app(app)
    i = 0
    if (string == "sala"):
        i = 1
    elif (string == "cozinha"):
        i = 2
    elif (string == "quarto"):
        i = 3
    elif (string == "banheiro"):
        i = 4
    var = Devices.query.filter_by(id=i).first()
    if(var.state == "False"):
        setattr(var, 'state', "True")
        db.session.commit()
    elif (var.state == "True"):
        setattr(var, 'state', "False")
        db.session.commit()
    return redirect("/")

#pagina para ler o status de uma tupla no banco de dados
@app.route("/state/<string>", methods=["POST", "GET"])
def sala_state(string):
    global db, app
    db.init_app(app)
    i = 0
    if(string == "sala"):
        i = 1
    elif(string == "cozinha"):
        i = 2
    elif (string == "quarto"):
        i = 3
    elif (string == "banheiro"):
        i = 4
    var = Devices.query.filter_by(id=i).first()
    return var.state


if __name__ == "__main__":
    app.run()
```

Código server:

```
import socket
import threading
import time
from get_html import *

SERVER_IP = socket.gethostbyname(socket.gethostname())
PORT = 7555
ADDR = (SERVER_IP, PORT)
FORMATO = 'utf-8'

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

conexoes = []
msg = ""

def get_comando():
    global msg
    while True:
        if msg == "":
            msg = get_html_f()

def handle_clientes(conn, addr):
    print(f"[NOVA CONEXAO] Um novo usuario se conectou pelo endereço {addr}")
    global conexoes
    global msg
    nome = "False"

    time.sleep(0.2)
    nome = conn.recv(1024).decode(FORMATO)
    mensagem_separada = nome.split("=")
    if len(mensagem_separada) > 1:
        nome = mensagem_separada[1]
    mapa_da_conexao = {
          "conn": conn,
          "addr": addr,
          "nome": nome,
          "status": False
    }
    conexoes.append(mapa_da_conexao)

    while(True):
        if msg.startswith(nome) and nome != "False":
              mensagem_separada = msg.split("=")
              if mensagem_separada[1]:
                    if mensagem_separada[1] == "True":
                          mapa_da_conexao["status"] = True
                    elif mensagem_separada[1] == "False":
                          mapa_da_conexao["status"] = False
                    conn.send(mensagem_separada[1].encode(FORMATO))
                    msg = ""
                    time.sleep(0.2)
        elif nome == "False":
            break
```

```
def start():
    print("[INICIANDO] Iniciando Socket")
    print(SERVER_IP)
    server.listen()
    thread2 = threading.Thread(target=get_comando, args=())
    thread2.start()
    while(True):
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_clientes, args=(conn, addr))
        thread.start()

start()
```

Código Get_url:

```
from bs4 import BeautifulSoup
from urllib3 import *

http = PoolManager()
url1 = "https://thebookisonthetable.herokuapp.com/state/sala"
url2 = "https://thebookisonthetable.herokuapp.com/state/cozinha"
url3 = "https://thebookisonthetable.herokuapp.com/state/quarto"
url4 = "https://thebookisonthetable.herokuapp.com/state/banheiro"
nome = []
nome.append("sala")
nome.append("cozinha")
nome.append("quarto")
nome.append("banheiro")
rawa = []
rawa.append("")
rawa.append("")
rawa.append("")
rawa.append("")

def get_html_f():
    global rawa
    html1 = http.request('GET', url1)
    html2 = http.request('GET', url2)
    html3 = http.request('GET', url3)
    html4 = http.request('GET', url4)
    raw = []
    raw.append(str(BeautifulSoup(html1.data, "html.parser")))
    raw.append(str(BeautifulSoup(html2.data, "html.parser")))
    raw.append(str(BeautifulSoup(html3.data, "html.parser")))
    raw.append(str(BeautifulSoup(html4.data, "html.parser")))
    resp = ""
    for i in range(4):
        if(rawa[i] != raw[i]):
            resp = nome[i] + "=" + raw[i]
            break
    rawa = raw
    return resp

url1s = "https://thebookisonthetable.herokuapp.com/comodos/sala"
url2s = "https://thebookisonthetable.herokuapp.com/comodos/cozinha"
```

```
url3s = "https://thebookisonthetable.herokuapp.com/comodos/quarto"
url4s = "https://thebookisonthetable.herokuapp.com/comodos/banheiro"

def set_html_f(name):
    global url1s, url2s, url3s, url4s
    if name == "sala":
        http.request('GET', url1s)
    if name == "cozinha":
        http.request('GET', url2s)
    if name == "quarto":
        http.request('GET', url3s)
    if name == "banheiro":
        http.request('GET', url4s)
```

Código Client:

```
import socket
import threading
import time

PORT = 7555
FORMATO = 'utf-8'
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)

class client():
    def __init__(self, nome):
        self.nome = nome
        self.cliente = False
        self.msg = ""
        self.state = False

    def enviar(self):
        self.cliente.send(self.msg.encode(FORMATO))

    def iniciar(self):
        self.cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.cliente.connect(ADDR)
        thread = threading.Thread(target=self.handle_mensagens, args=())
        thread.start()

    def handle_mensagens(self):
         self.msg = "nome" + "=" + self.nome
         print(self.msg)
         self.enviar()
         while(True):
             if(len(self.nome) > 1):
                 self.msg = self.cliente.recv(1024).decode()
                 if(self.msg == "True"):
                     self.state = True
                     print(self.nome + "=" + self.msg)
                 elif(self.msg == "False"):
                     self.state = False
                     print(self.nome + "=" + self.msg)
             else:
                 break
```

```
sala = client("sala")
cozinha = client("cozinha")
quarto = client("quarto")
banheiro = client("banheiro")

time.sleep(2)
sala.iniciar()
time.sleep(2)
cozinha.iniciar()
time.sleep(2)
quarto.iniciar()
time.sleep(2)
banheiro.iniciar()
```

Código interface:

```
from kivy.app import App
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from kivy.uix.image import Image
from kivy.clock import *
from get_html import *
from client import *


class app_casa(App):
    def build(self):

        global sala, cozinha, quarto, banheiro

        #returns a window object with all it's widgets
        self.window = GridLayout()
        self.window.cols = 2
        self.window.size_hint = (1, 0.8)
        self.window.pos_hint = {"center_x": 0.5, "center_y": 0.5}

        self.light_sala = Image(source="images/sala_off.png")
        self.window.add_widget(self.light_sala)

        self.light_cozinha = Image(source="images/cozinha_off.png")
        self.window.add_widget(self.light_cozinha)

        self.button_sala = Button(
            text="sala",
            size_hint=(0.5, 0.2),
            bold=True,
            background_color='#000000'
            # remove darker overlay of background colour
            # background_normal = ""
        )
        self.button_sala.bind(on_press=self.sala_set_light)
        self.window.add_widget(self.button_sala)

        self.button_cozinha = Button(
            text="cozinha",
```

```
            # size_hint=(1, 0.5),
            size_hint=(0.5, 0.2),
            bold=True,
            background_color='#000000',
            # remove darker overlay of background colour
            # background_normal = ""
        )
        self.button_cozinha.bind(on_press=self.cozinha_set_light)
        self.window.add_widget(self.button_cozinha)

        self.light_quarto = Image(source="images/quarto_off.png")
        self.window.add_widget(self.light_quarto)

        self.light_banheiro = Image(source="images/banheiro_off.png")
        self.window.add_widget(self.light_banheiro)

        self.button_quarto = Button(
            text="quarto",
            # size_hint=(1, 0.5),
            size_hint=(0.5, 0.2),
            bold=True,
            background_color='#000000'
            # remove darker overlay of background colour
            # background_normal = ""
        )
        self.button_quarto.bind(on_press=self.quarto_set_light)
        self.window.add_widget(self.button_quarto)

        self.button_banheiro = Button(
            text="banheiro",
            #size_hint=(1, 0.5),
            size_hint=(0.5, 0.2),
            bold=True,
            background_color='#000000'
            # remove darker overlay of background colour
            # background_normal = ""
        )
        self.button_banheiro.bind(on_press=self.banheiro_set_light)
        self.window.add_widget(self.button_banheiro)

        Clock.schedule_interval(self.mainloop, 0.05)

        return self.window

    def light_on(self, nome):
        if nome == "sala":
            self.light_sala.set_texture_from_resource("images/sala_on.png")
        if nome == "cozinha":

self.light_cozinha.set_texture_from_resource("images/cozinha_on.png")
        if nome == "quarto":

self.light_quarto.set_texture_from_resource("images/quarto_on.png")
        if nome == "banheiro":

self.light_banheiro.set_texture_from_resource("images/banheiro_on.png")

    def light_off(self, nome):
```

```
        if nome == "sala":
            self.light_sala.set_texture_from_resource("images/sala_off.png")
        if nome == "cozinha":

self.light_cozinha.set_texture_from_resource("images/cozinha_off.png")
        if nome == "quarto":

self.light_quarto.set_texture_from_resource("images/quarto_off.png")
        if nome == "banheiro":

self.light_banheiro.set_texture_from_resource("images/banheiro_off.png")

    def sala_set_light(self, instance):
        set_html_f("sala")

    def cozinha_set_light(self, instance):
        set_html_f("cozinha")

    def quarto_set_light(self, instance):
        set_html_f("quarto")

    def banheiro_set_light(self, instance):
        set_html_f("banheiro")

    def mainloop(self, instance):
        if sala.state:
            self.light_on("sala")
        elif not sala.state:
            self.light_off("sala")
        if cozinha.state:
            self.light_on("cozinha")
        elif not cozinha.state:
            self.light_off("cozinha")
        if quarto.state:
            self.light_on("quarto")
        elif not quarto.state:
            self.light_off("quarto")
        if banheiro.state:
            self.light_on("banheiro")
        elif not banheiro.state:
            self.light_off("banheiro")
app = app_casa()
app.run()
```