
MINISTRY OF EDUCATION AND TRAINING

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY OF HIGH QUALITY TRAINING

-----🕮-----

FINAL-TERM COURSE REPORT

INTERNET OF THINGS

SMART HOME

Lecturer: MSc Bui Ha Duc

SVE: Nguyen Ngoc Quy – 20146056

Major: Mechatronics Engineering

Ho Chi Minh City, May 30, 2023

Contents

INTRODUCTION

1. Background

The Internet of Things (IoT) has emerged as a groundbreaking technology in recent years, captivating widespread attention for its ability to connect and control diverse devices through the internet. This revolutionary concept holds the promise of transforming our interactions with the world around us and streamlining everyday tasks. At the heart of IoT lies the power to remotely monitor and control a vast array of devices, including sensors and actuators, using intuitive web applications.

IoT has ushered in a new era of connectivity, enabling seamless communication between devices, humans, and the digital realm. Through this interconnectedness, our surroundings become smarter and more responsive, offering enhanced convenience, efficiency, and productivity. With IoT, we can effortlessly monitor our homes' security systems, adjust the temperature and lighting remotely, and even receive notifications about the freshness of our groceries—all through the convenience of web applications.

Imagine a world where you can effortlessly manage your energy consumption, optimize your daily routines, and remotely control devices from anywhere. IoT empowers us to achieve this level of control and convenience. By seamlessly integrating sensors and actuators into our homes, workplaces, and cities, IoT enables us to gather real-time data and make informed decisions.

Web applications serve as the gateway to this interconnected ecosystem. They provide us with user-friendly interfaces to interact with our IoT devices, empowering us to monitor their status, change settings, and receive valuable insights. Whether it's adjusting the thermostat to the perfect temperature before arriving home, remotely unlocking the front door for a trusted guest, or receiving alerts about potential water leaks, web applications enable us to stay connected and in control.

Moreover, IoT extends its reach beyond personal convenience. It holds immense potential for industries and businesses, revolutionizing sectors such as healthcare, transportation, agriculture, and manufacturing. In healthcare, IoT devices enable remote patient monitoring, ensuring timely intervention and personalized care. In transportation, IoT technology enables smart traffic management systems, optimizing routes and reducing congestion. In agriculture, IoT sensors help monitor soil moisture levels and environmental conditions, ensuring optimal crop growth. And in manufacturing, IoT-powered systems facilitate predictive maintenance, reducing downtime and improving efficiency.

However, along with the tremendous benefits IoT offers, there are also considerations regarding privacy, security, and data management. With increased connectivity and data exchange, robust measures must be in place to safeguard personal information and prevent unauthorized access. Moreover, effective data management practices and regulations are crucial to ensure the responsible and ethical use of the vast amounts of data generated by IoT devices.

As IoT continues to advance, it is essential for individuals, businesses, and policymakers to collaborate and address these challenges while embracing the transformative potential of this technology. By doing so, we can unlock a future where connectivity, efficiency, and convenience are seamlessly woven into the fabric of our daily lives.

In conclusion, the Internet of Things has emerged as a game-changing technology, enabling the connection and control of devices through web applications. Its transformative potential spans across various sectors and has the capacity to enhance our lives in numerous ways. As we navigate this IoT-powered future, it is crucial to prioritize privacy, security, and responsible data management, ensuring that we unlock the full benefits while mitigating potential risks. By embracing IoT and harnessing its potential, we pave the way for a smarter, more connected world.

2. Objectives

The primary purpose of this report is to outline the implementation of an IoT device control system based on Flask and designed for the Raspberry Pi platform. This system leverages provided code to enable remote control of sensors and devices through a web application. Our main objective in developing this solution is to create a versatile and user-friendly IoT system utilizing technologies like Flask, Adafruit_DHT, and RPi.GPIO.

With the rapid expansion of IoT, there is a growing need for practical and accessible solutions that allow users to interact with their devices remotely. By employing Flask, a popular Python web framework, we aim to develop an intuitive and efficient web application for controlling IoT devices. Flask provides a robust and flexible foundation, enabling us to create a seamless user experience.

To interface with the physical world, we utilize the Adafruit_DHT library, which offers support for DHT11 temperature and humidity sensors. By integrating this library into our system, we can accurately monitor environmental conditions and gather data for further analysis. This functionality expands the capabilities of our IoT solution, allowing users to remotely monitor and react to changes in temperature and humidity.

Furthermore, the RPi.GPIO library plays a pivotal role in our implementation. It facilitates the control of GPIO pins on the Raspberry Pi, enabling the manipulation of actuators and the interaction with various electronic components. By utilizing the RPi.GPIO library, we can extend our IoT system beyond mere monitoring and incorporate the ability to remotely control devices such as lights, fans, and even TVs.

The overarching goal of this project is to create a streamlined and user-friendly IoT device control system that seamlessly integrates with the Raspberry Pi platform. We aim to provide a comprehensive guide that enables users to understand the implementation process and leverage the provided code to build upon and customize the system according to their specific needs.

Through this project, we aspire to empower individuals and organizations to harness the potential of IoT and leverage its benefits in their daily lives and operations. By creating a simple yet flexible IoT solution, we hope to inspire innovation and creativity in the field of IoT device control.

In conclusion, the objectives of this report revolve around the implementation of a Flask-based IoT device control system on the Raspberry Pi platform. By utilizing Flask, Adafruit_DHT, and RPi.GPIO, our aim is to develop a user-friendly and versatile solution that enables remote control and monitoring of sensors and devices through a web application. We hope that this project will serve as a stepping stone for further exploration and utilization of IoT technologies in various domains.

SMART HOME

1. Introduction

In this project, we are presented with a codebase that implements a Flask application designed to control a range of devices and sensors using a Raspberry Pi. The application offers users the ability to unlock and lock a door, control the lighting in different rooms, manage a fan and TV, as well as retrieve data from various sensors including an infrared sensor, motion sensor, and DHT11 temperature and humidity sensor. Through the use of Flask routes, users can interact with these features, while the application also provides real-time sensor data using server-sent events.

The goal of this project is to create a user-friendly interface that enables efficient control and monitoring of connected devices and sensors. By leveraging the Flask framework, the code simplifies the process of developing a web-based platform for users to interact with the Raspberry Pi. This approach ensures ease of use and accessibility for individuals with varying levels of technical expertise.

To establish communication between the application and the Raspberry Pi's hardware, the code configures the GPIO (General Purpose Input Output) pins. Each device and sensor is assigned specific GPIO configurations, including the SR505 motion sensor, DHT11 sensor, automatic lock, lighting systems, fan, TV, and infrared sensor. By configuring these pins, the code enables seamless integration and control of the connected devices.

Threading is employed to continuously monitor the sensor values in the background. By running separate threads for each sensor, the application can continually update and provide accurate real-time data to the users. This ensures that users have access to the latest information from the sensors, facilitating informed decision-making and enhancing the overall user experience.

The Flask routes defined in the code serve as endpoints to handle different user requests. For example, users can unlock and lock the door through dedicated routes, control the lighting in specific rooms, adjust the fan or TV settings, and retrieve real-time sensor data. The defined routes ensure that the appropriate actions are executed based on user inputs, enabling seamless control and monitoring of the connected devices and sensors.

Furthermore, the application utilizes server-sent events to deliver real-time sensor data to the users. By generating and streaming data using these events, users can receive live updates from the sensors without the need for constant refreshing of the web page. This feature enhances the responsiveness of the application and provides users with up-to-date information.

In conclusion, the provided code demonstrates a comprehensive Flask application that effectively controls various devices and sensors using a Raspberry Pi. Through the use of Flask routes, GPIO configurations, threading, and server-sent events, the application offers users a user-friendly interface to interact with the connected devices and access real-time sensor data. This project showcases the power and versatility of the Flask framework and highlights the potential for creating robust and interactive IoT (Internet of Things) applications using the Raspberry Pi platform.

2. Methodology

The methodology employed in this project centers around the utilization of two key components: the Flask framework and the GPIO library. These elements play crucial roles in facilitating communication between the application and the Raspberry Pi's pins and devices. By leveraging the Flask framework, developers are able to create a web-based interface through which users can interact with the system.

To ensure seamless connectivity with the hardware, the code establishes essential GPIO configurations for each specific device and sensor. These include the SR505 motion sensor, DHT11 sensor, automatic lock, lights, fan, TV, and infrared sensor. By configuring the GPIO pins accordingly, the system can effectively communicate with these components, enabling control and data acquisition.

In order to monitor the values of the various sensors in real-time, threading is implemented. This technique allows for continuous background monitoring, ensuring that the system remains responsive and up-to-date with the current sensor readings. Consequently, users can receive accurate and timely data from the sensors, enhancing the overall functionality and reliability of the application.

To facilitate user interaction, Flask routes are defined within the code. These routes act as endpoints for handling different types of requests initiated by the user. Whether it's unlocking or locking the door, controlling lights in different rooms, adjusting the fan or TV settings, or accessing real-time sensor data, the Flask routes ensure that the appropriate actions are performed based on the user's input. This modular approach to routing streamlines the user experience and enhances the overall usability of the system.

By combining the Flask framework, GPIO library, threading, and well-defined routes, this project successfully achieves its objectives of providing a robust and user-friendly interface for controlling and monitoring the Raspberry Pi's connected devices and sensors. The methodology employed in this project lays the foundation for a seamless and efficient interaction between users and the underlying hardware, paving the way for a wide range of applications and functionalities.

3. Implementation

The implementation of the project begins by initializing the Flask application and importing the required modules and libraries. Flask provides the foundation for creating the web-based interface that allows users to interact with the Raspberry Pi and its connected devices.

The code proceeds to set up the GPIO pin assignments and configurations for the various devices and sensors. This step ensures that the Raspberry Pi can effectively communicate with and control these components. The GPIO pins are initialized and configured based on the specific requirements of each device, such as the SR505 motion sensor, DHT11 sensor, automatic lock, lights in different rooms, fan, TV, and infrared sensor.

To continuously monitor the sensor values in the background, separate threads are created. These threads are responsible for reading data from the motion sensor, SR505 sensor, and DHT11 sensor at regular intervals. By running these tasks in parallel, the application can provide real-time updates and accurate readings from the sensors, enhancing the responsiveness and reliability of the system.

The code also defines functions to control the automatic lock, lights in different rooms, fan, and TV. These functions allow users to interact with and manipulate the connected devices based on their preferences. For example, the lock_door() and unlock_door() functions control the automatic lock, while the control_light()

function enables users to turn the lights on or off in specific rooms. Similarly, the control_quat() and control_tv() functions control the fan and TV respectively.

To handle user requests, the code implements Flask routes. These routes define the URLs and HTTP methods for different functionalities. For instance, the /unlock route handles the request to unlock the door, while the /lock route handles the request to lock the door. The /light, /fan, and /tv routes handle requests to control the lights, fan, and TV respectively. Additionally, the /sensor_data, /sr505_data, and /dht11_data routes provide real-time sensor data to users through server-sent events.

The server-sent events enable the application to send data from the sensors to the client-side in real-time. By utilizing the Response object in Flask, the code establishes a continuous connection with the client and streams the sensor data as events. This feature ensures that users can access the latest readings from the sensors without the need for manual refreshes, enhancing the overall user experience.

In summary, the implementation of the project involves initializing the Flask application, configuring GPIO pin assignments, creating separate threads for sensor data monitoring, defining functions for device control, implementing Flask routes to handle user requests, and utilizing server-sent events to provide real-time sensor data. This comprehensive implementation allows users to seamlessly interact with the Raspberry Pi and its connected devices, enabling efficient control and monitoring capabilities.

i. Main code

```
from flask import Flask, request, render_template, Response
import Adafruit_DHT
import requests
import RPi.GPIO as GPIO
import time
import threading
import json
from flask import redirect, url_for

app = Flask(__name__)

raspberry_ip = '172.20.10.2'  # Địa chỉ IP của Raspberry Pi

# Khai báo GPIO cảm biến SR505
SR505_pin = 19

# Khai báo GPIO cảm biến DHT11
DHT11_pin = 17
```

```

# Khai báo các GPIO cho khóa tự động
lock_pin = 18
password = "1234"  # Mật khẩu cần nhập

# Khai báo các GPIO cho đèn
kitchen_light_pin = 23
bedroom_light_pin = 24
bathroom_light_pin = 25
living_room_light_pin = 26

# Khai báo GPIO cho thiết bị
quat_pin = 12
tv_pin = 13

# Khai báo GPIO cho cảm biến hồng ngoại
led_pin = 20
lm393_pin = 27
led_pin_1 = 21

# Thiết lập chế độ chân GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
GPIO.setup(led_pin_1, GPIO.OUT)
GPIO.setup(lm393_pin, GPIO.IN)

# Thiết lập GPIO cho khóa tự động
GPIO.setup(lock_pin, GPIO.OUT)

# Thiết lập GPIO cho đèn
GPIO.setup(kitchen_light_pin, GPIO.OUT)
GPIO.setup(bedroom_light_pin, GPIO.OUT)
GPIO.setup(bathroom_light_pin, GPIO.OUT)
GPIO.setup(living_room_light_pin, GPIO.OUT)

# Thiết lập GPIO cho thiết bị
GPIO.setup(quat_pin, GPIO.OUT)
GPIO.setup(tv_pin, GPIO.OUT)

# Thiết lập GPIO cho cảm biến
GPIO.setup(SR505_pin, GPIO.IN)

# Thiết lập GPIO cho cảm biến DHT11
DHT11_sensor = Adafruit_DHT.DHT11

def unlock_door():
    GPIO.output(lock_pin, GPIO.HIGH)
    print("Cửa đã mở")

def lock_door():
    GPIO.output(lock_pin, GPIO.LOW)
    print("Cửa đã khóa")

```

```
   def control_light(pin, state):
       GPIO.output(pin, GPIO.HIGH if state == 'on' else GPIO.LOW)

   def control_quat(pin, state):
       GPIO.output(pin, GPIO.HIGH if state == 'on' else GPIO.LOW)

   def control_tv(pin, state):
    GPIO.output(pin, GPIO.HIGH if state == 'on' else GPIO.LOW)

   def control_led():
       GPIO.output(led_pin, GPIO.HIGH if GPIO.input(lm393_pin) == GPIO.LOW else GPIO.LOW)

   def sensor_thread():
       lm393_data = []
       while True:
                 control_led()
           lm393_value = GPIO.input(lm393_pin)
        lm393_data.append(lm393_value)
        time.sleep(0.1)

   sensor_thread = threading.Thread(target=sensor_thread)
sensor_thread.daemon = True
sensor_thread.start()

   def sensor_thread1():
       SR505_data = []
    while True:
        SR505_value = GPIO.input(SR505_pin)
        SR505_data.append(SR505_value)
        time.sleep(0.1)

sensor_thread1 = threading.Thread(target=sensor_thread1)
sensor_thread1.daemon = True
sensor_thread1.start()

   def sensor_thread2():
    DHT11_data = []
    while True:
        humidity, temperature = Adafruit_DHT.read_retry(DHT11_sensor, DHT11_pin)
        if humidity is not None and temperature is not None:
               data = {'humidity': humidity, 'temperature': temperature}
            DHT11_data.append(data)
        time.sleep(2)

   sensor_thread2 = threading.Thread(target=sensor_thread2)
   sensor_thread2.daemon = True
sensor_thread2.start()

@app.route('/')
   def home():
       return render_template('index.html')
```

```

@app.route('/unlock', methods=['POST', 'GET'])
   def unlock():
       if request.method == 'POST':
           entered_password = request.form['password']
           if entered_password == password:
            unlock_door()
               requests.get(f'http://{raspberry_ip}/unlock')
               return "MỞ CỬA"
           else:
               return "Mật khẩu không chính xác"
       return render_template('unlock.html')

   @app.route('/lock')
         def lock():
       lock_door()
    requests.get(f'http://{raspberry_ip}/lock')
    return "KHÓA CỬA"

   @app.route('/light', methods=['POST'])
   def control_light_route():
    room = request.form['room']
       state = request.form['state']
       pin = None

       if room == 'kitchen':
        pin = kitchen_light_pin
    elif room == 'bedroom':
        pin = bedroom_light_pin
       elif room == 'bathroom':
        pin = bathroom_light_pin
    elif room == 'living_room':
        pin = living_room_light_pin
       elif room == 'car':
           pin = led_pin_1

    if pin is not None:
        control_light(pin, state)

       return 'Đã điều khiển đèn'

@app.route('/all_lights_on')
   def all_lights_on():
       control_light(kitchen_light_pin, 'on')
       control_light(bedroom_light_pin, 'on')
       control_light(bathroom_light_pin, 'on')
    control_light(living_room_light_pin, 'on')
       control_light(led_pin_1, 'on')
       return 'Đã bật tất cả đèn'

   @app.route('/all_lights_off')
   def all_lights_off():
       control_light(kitchen_light_pin, 'off')
```

```python
    control_light(bedroom_light_pin, 'off')
    control_light(bathroom_light_pin, 'off')
    control_light(living_room_light_pin, 'off')
    control_light(led_pin_1, 'off')
    return 'Đã tắt tất cả đèn'

@app.route('/fan', methods=['POST'])
def control_fan_route():
    state = request.form['state']
    control_quat(quat_pin, state)
    return 'Đã điều khiển quạt'

@app.route('/tv', methods=['POST'])
def control_tv_route():
    state = request.form['state']
    control_quat(tv_pin, state)
    return 'Đã điều khiển tv'

@app.route('/sensor_data')
def sensor_data():
    def generate():
        while True:
            lm393_value = GPIO.input(lm393_pin)
            data = {'label': str(time.time()), 'value': lm393_value}
            yield f"data: {json.dumps(data)}\n\n"
            time.sleep(0.1)

    return Response(generate(), mimetype='text/event-stream')

@app.route('/sr505_data')
def sr505_data():
    def generate1():
        while True:
            SR505_value = GPIO.input(SR505_pin)
            data1 = {'label': str(time.time()), 'value': SR505_value}
            yield f"data: {json.dumps(data1)}\n\n"
            time.sleep(0.1)

    return Response(generate1(), mimetype='text/event-stream')

@app.route('/dht11_data')
def dht11_data():
    def generate2():
        while True:
            humidity, temperature = Adafruit_DHT.read_retry(DHT11_sensor, DHT11_pin)
            if humidity is not None and temperature is not None:
                data2 = {'label': str(time.time()), 'humidity': humidity, 'temperature': temperature}
                yield f"data: {json.dumps(data2)}\n\n"
            time.sleep(2)

    return Response(generate2(), mimetype='text/event-stream')

if __name__ == '__main__':
    app.run(host='172.20.10.2', port=8000)

```

ii. Website Code

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home</title>
    <style>
        body {
            background-image:
url('https://images.pexels.com/photos/1743165/pexels-photo-
1743165.jpeg?auto=compress&cs=tinysrgb&w=1600');
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
        }
        h1 {
            color: #ffffff;
            text-align: center;
        }
        p {
            text-align: center;
        }
        .button {
            display: inline-block;
            background-color: #45a049;
            color: #ffffff;
            text-decoration: none;
            padding: 10px 20px;
            margin-top: 10px;
            border-radius: 5px;
        }

        .button:hover {
            background-color: #ff0000;
        }

        .clock {
            text-align: center;
            margin-top: 20px;
            font-size: 24px;
            color: #ffffff;
        }

        .chart-container {
            width: 700px;
            height: 400px;
            margin: 20px;
            background-image: url('https://images.unsplash.com/photo-
1501630834273-4b5604d2ee31?ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8OXx8Y2xvdWR8ZW58MHx8MHx8fDA%3D&auto=for
mat&fit=crop&w=500&q=60');
            background-size: cover; /* Tự động điều chỉnh kích thước phù
hợp */
        }

        .chart-container1 {
            width: 700px;
            height: 400px;
            margin: 20px;
            background-image: url('https://images.unsplash.com/photo-
1501630834273-4b5604d2ee31?ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8OXx8Y2xvdWR8ZW58MHx8MHx8fDA%3D&auto=for
mat&fit=crop&w=500&q=60');
            background-size: cover; /* Tự động điều chỉnh kích thước phù
hợp */
        }

        .chart-wrapper {
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .chart-container2 {
            width: 700px;
            height: 400px;
            margin: 20px;
            background-image: url('https://images.unsplash.com/photo-
1501630834273-4b5604d2ee31?ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8OXx8Y2xvdWR8ZW58MHx8MHx8fDA%3D&auto=for
mat&fit=crop&w=500&q=60');
            background-size: cover; /* Tự động điều chỉnh kích thước phù
hợp */
        }
        .chart-wrapper1 {
            display: flex;
            justify-content: center;
            align-items: center;
        }

        /* Thêm phần CSS cho bảng */
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }

        th, td {
            border: 1px solid #ffffff;
            padding: 8px;
            color: #ffffff;
            text-align: center;
        }

        th {
            background-color: #005f08;
        }

        td {
            background-color: #26aac2;
        }

        td.on {
            background-color: #00ff00;
        }

        td.off {
            background-color: #ff0000;
            color: #ffffff;
        }
        .highlight {
            color: white; /* Màu chữ trắng */
            font-weight: bold; /* Chữ đậm */
        }
        .sidebar {
            height: 100%;
            width: 200px;
            position: fixed;
            z-index: 1;
            top: 0;
            left: 0;
            background-color: #111;
            overflow-x: hidden;
            padding-top: 20px;
        }

        .sidebar a {
            padding: 8px 8px 8px 16px;
            text-decoration: none;
            font-size: 20px;
            color: #818181;
            display: block;
        }

        .sidebar a:hover {
            color: #f1f1f1;
        }

        /* CSS cho nội dung chính */
        .content {
            margin-left: 200px;
            padding: 20px;
        }

        .page {
            display: none;
        }

        .page.active {
            display: block;
        }
    </style>
    <!-- Thêm thư viện Chart.js -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script>
        function controlLight(room, state) {
            fetch('/light', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: 'room=' + encodeURIComponent(room) + '&state=' +
encodeURIComponent(state)
            })
            .then(response => {
                if (response.ok) {
                    console.log('Đã gửi yêu cầu điều khiển đèn');
                    updateStatus(room, state);
                } else {
                    console.error('Lỗi khi gửi yêu cầu điều khiển đèn');
                }
            })
            .catch(error => {
                console.error('Lỗi khi gửi yêu cầu điều khiển đèn:',
error);
            });
        }

        function updateStatus(room, status) {
            const statusCell = document.getElementById(room + '_status');
            statusCell.textContent = status;
            statusCell.className = status === 'on' ? 'on' : 'off';
        }

        function turnOnAllLights() {
            fetch('/all_lights_on', {
                method: 'GET'
            })
            .then(response => {
                if (response.ok) {
                    console.log('Đã gửi yêu cầu bật tất cả đèn');
                    updateAllLightsStatus('on');
                } else {
                    console.error('Lỗi khi gửi yêu cầu bật tất cả đèn');
                }
            })
            .catch(error => {
                console.error('Lỗi khi gửi yêu cầu bật tất cả đèn:',
error);
            });
        }

        function turnOffAllLights() {
            fetch('/all_lights_off', {
                method: 'GET'
            })
            .then(response => {
                if (response.ok) {
                    console.log('Đã gửi yêu cầu tắt tất cả đèn');
                    updateAllLightsStatus('off');
                } else {
                    console.error('Lỗi khi gửi yêu cầu tắt tất cả đèn');
                }
            })
            .catch(error => {
                console.error('Lỗi khi gửi yêu cầu tắt tất cả đèn:',
error);
            });
        }

        function updateAllLightsStatus(status) {
            const livingRoomStatus =
document.getElementById('living_room_status');
            const kitchenStatus =
document.getElementById('kitchen_status');
            const bedroomStatus =
document.getElementById('bedroom_status');
            const bathroomStatus =
document.getElementById('bathroom_status');
            const allLightsStatus =
document.getElementById('all_lights_status');
            const carStatus = document.getElementById('car_status');
            livingRoomStatus.textContent = status;
            livingRoomStatus.className = status === 'on' ? 'on' : 'off';

            kitchenStatus.textContent = status;
            kitchenStatus.className = status === 'on' ? 'on' : 'off';

            bedroomStatus.textContent = status;
            bedroomStatus.className = status === 'on' ? 'on' : 'off';

            bathroomStatus.textContent = status;
            bathroomStatus.className = status === 'on' ? 'on' : 'off';

            allLightsStatus.textContent = status;
            allLightsStatus.className = status === 'on' ? 'on' : 'off';

            carStatus.textContent = status;
            carStatus.className = status === 'on' ? 'on' : 'off';
        }

        function controlFan(state) {
            fetch('/fan', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded'
                },
                body: `state=${state}`
            })
            .then(response => {
                if (response.ok) {
                    console.log('Đã gửi yêu cầu điều khiển quạt');
                    updateFanStatus(state);
                } else {
                    console.error('Lỗi khi gửi yêu cầu điều khiển quạt');
                }
            })
            .catch(error => {
                console.error('Lỗi khi gửi yêu cầu điều khiển quạt:',
error);
            });
        }

        function updateFanStatus(status) {
            const fanStatus = document.getElementById('fan_status');
            fanStatus.textContent = status;
            fanStatus.className = status === 'on' ? 'on' : 'off';
        }

        function controlTV(state) {
            fetch('/tv', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded'
                },
                body: `state=${state}`
            })
            .then(response => {
                if (response.ok) {
                    console.log('Đã gửi yêu cầu điều khiển tv');
                    updateTVStatus(state);
                } else {
                    console.error('Lỗi khi gửi yêu cầu điều khiển tv');
                }
            })
            .catch(error => {
                console.error('Lỗi khi gửi yêu cầu điều khiển tv:', error);
            });
        }

        function updateTVStatus(status) {
            const tvStatus = document.getElementById('tv_status');
            tvStatus.textContent = status;
            tvStatus.className = status === 'on' ? 'on' : 'off';
        }

        function showPage(pageName) {
            // Ẩn tất cả các trang
            var pages = document.getElementsByClassName("page");
            for (var i = 0; i < pages.length; i++) {
                pages[i].style.display = "none";
            }

            // Hiển thị trang được chọn
            document.getElementById(pageName).style.display = "block";
        }
    </script>
</head>
<body>
    <div class="sidebar">
        <a href="javascript:void(0)" onclick="showPage('home')">Trang
chủ</a>
        <a href="javascript:void(0)" onclick="showPage('lights')">Điều
khiển đèn</a>
        <a href="javascript:void(0)" onclick="showPage('fan')">Điều khiển
thiết bị</a>
        <a href="javascript:void(0)" onclick="showPage('sensor')">Tình
trạng ngôi nhà</a>
    </div>
    <div class="content">
        <!-- Trang chủ -->
        <div id="home" class="page">
            <h1><span style="color: white;">Điều khiển khóa cửa</span></h1>
            <p><a href="/unlock" class="button"
onclick="incrementCounter()"><span style="color: black;">Mở
cửa</span></a></p>
            <p><a href="/lock" class="button"
onclick="resetCounter()"><span style="color: black;">Khóa
cửa</span></a></p>
            <!-- Nội dung trang chủ -->
            <div class="youtube-video" style="text-align: center;">
                <iframe width="1024" height="768"
src="https://www.youtube.com/embed/XiLY3UqKcBo" frameborder="0"
allowfullscreen></iframe>
            </div>
        </div>

        <!-- Trang điều khiển đèn -->
        <div id="lights" class="page" style="display: none;">
            <h2><span style="color: white;">Điều khiển đèn</span></h2>
            <!-- Nội dung điều khiển đèn -->
            <!-- Bảng điều khiển đèn -->
            <table>
                <tr>
```

```
                    <th>Số thứ tự</th>
                    <th>Led</th>
                    <th>Bật/Tắt</th>
                    <th>Trạng thái</th>
                </tr>
                <tr>
                    <td>1</td>
                    <td>Phòng khách</td>
                    <td>
                        <button onclick="controlLight('living_room',
'on')">Bật</button>
                        <button onclick="controlLight('living_room',
'off')">Tắt</button>
                    </td>
                    <td id="living_room_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>Nhà bếp</td>
                    <td>
                        <button onclick="controlLight('kitchen',
'on')">Bật</button>
                        <button onclick="controlLight('kitchen',
'off')">Tắt</button>
                    </td>
                    <td id="kitchen_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>3</td>
                    <td>Phòng ngủ</td>
                    <td>
                        <button onclick="controlLight('bedroom',
'on')">Bật</button>
                        <button onclick="controlLight('bedroom',
'off')">Tắt</button>
                    </td>
                    <td id="bedroom_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>4</td>
                    <td>Phòng tắm</td>
                    <td>
                        <button onclick="controlLight('bathroom',
'on')">Bật</button>
                        <button onclick="controlLight('bathroom',
'off')">Tắt</button>
                    </td>
                    <td id="bathroom_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>5</td>
                    <td>Nhà xe</td>
                    <td>
```

```
                        <button onclick="controlLight('car',
'on')">Bật</button>
                        <button onclick="controlLight('car',
'off')">Tắt</button>
                    </td>
                    <td id="car_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>6</td>
                    <td>Tất cả</td>
                    <td>
                        <button onclick="turnOnAllLights()">Bật</button>
                        <button onclick="turnOffAllLights()">Tắt</button>
                    </td>
                    <td id="all_lights_status" class="off">Wait</td>
                </tr>

            </table>
        </div>
        <!-- Trang điều khiển quạt -->
        <div id="fan" class="page">
            <h2><span style="color: white;">Điều khiển thiết bị</span></h2>
            <!-- Nội dung điều khiển thiết bị -->
            <table>
                <tr>
                    <th>Số thứ tự</th>
                    <th>Thiết bị</th>
                    <th>Bật/Tắt</th>
                    <th>Trạng thái</th>
                </tr>
                <tr>
                    <td>1</td>
                    <td>Quạt</td>
                    <td>
                        <button onclick="controlFan('on')">Bật</button>
                        <button onclick="controlFan('off')">Tắt</button>
                    </td>
                    <td id="fan_status" class="off">Wait</td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>TV</td>
                    <td>
                        <button onclick="controlTV('on')">Bật</button>
                        <button onclick="controlTV('off')">Tắt</button>
                    </td>
                    <td id="tv_status" class="off">Wait</td>
                </tr>
            </table>
        </div>
        <div id="sensor" class="page">
            <h3><span style="color: white;">Tình trạng ngôi
nhà</span></h3>
```

```
            <!-- Nội dung tình trạng ngôi nhà -->
            <div class="chart-wrapper">
                <div class="chart-container">
                    <canvas id="chart"></canvas>
                </div>
                <div class="chart-container1">
                    <canvas id="chart1"></canvas>
                </div>
            </div>
            <div style="display: flex; justify-content: center;">
                <div style="overflow-y: auto; max-height: 200px; margin-
right: 15px; width: 700px;">
                    <p class="highlight" style="font-size: 18px; padding-
bottom: 10px;">Bảng thông số cảm biến NHÀ XE</p>
                    <table style="font-size: 20px;">
                        <tr>
                            <th style="padding: 10px;">Thời gian</th>
                            <th style="padding: 10px;">Thông tin</th>
                        </tr>
                        <tbody id="sensor_data"></tbody>
                    </table>
                </div>
                <div style="overflow-y: auto; max-height: 200px; margin-
left: 15px; width: 700px;">
                    <p class="highlight" style="font-size: 18px; padding-
bottom: 10px;">Bảng thông số cảm biến CỬA</p>
                    <table style="font-size: 20px;">
                        <tr>
                            <th style="padding: 10px;">Thời gian</th>
                            <th style="padding: 10px;">Thông tin</th>
                        </tr>
                        <tbody id="sr505_data"></tbody>
                    </table>
                </div>
            </div>


            <script>
                // Lấy tham chiếu đến phần tử tbody
                const sensorDataContainer =
document.getElementById('sensor_data');

                // Mảng lưu trữ dữ liệu cảm biến
                const sensorData = [];

                // Khởi tạo biểu đồ
                const ctx =
document.getElementById('chart').getContext('2d');
                const chart = new Chart(ctx, {
                type: 'line',
                data: {
                    labels: [],
                    datasets: [{
                        label: 'Giá trị cảm biến hồng ngoại',
                        data: [],
                        backgroundColor: 'rgba(255, 255, 255, 0.2)', //
Màu nền trắng

                        borderColor: 'rgba(255, 0, 0, 1)', // Màu đỏ
                        borderWidth: 5,
                        fill: true,
                    }]
                },
                options: {
                    scales: {
                        x: {
                            display: true,
                            grid: {
                                color: 'white' // Màu cột x trắng
                            },
                            ticks: {
                                color: 'darkgreen', // Màu thước đo y xanh
đậm

                                font: {
                                    size: 20, // Kích thước chữ to hơn
(14px)

                                    weight: 'bold' // Đậm
                                }
                            }
                        },

                        y: {
                            display: true,
                            suggestedMin: 0,
                            suggestedMax: 1,
                            stepSize: 1,
                            grid: {
                                color: 'blue' // Màu cột y trắng
                            },
                            ticks: {
                                color: 'darkgreen', // Màu thước đo y xanh
đậm

                                font: {
                                    size: 20, // Kích thước chữ to hơn
(14px)

                                    weight: 'bold' // Đậm
                                }
                            }
                        }
                    },
                    plugins: {
                        legend: {
                            labels: {
                                color: 'Black' // Màu chữ trắng
                            }
                        }
                    }
                }
            });


                // Lưu trữ trạng thái hiện tại của cảm biến hồng ngoại
                let currentStatus = '';

                // Lắng nghe sự kiện SSE và cập nhật dữ liệu cảm biến hồng
ngoại
                const eventSource = new EventSource('/sensor_data');
                eventSource.addEventListener('message', function(event) {
                    const data = JSON.parse(event.data);
                    const label = new Date(parseFloat(data.label) *
1000).toLocaleTimeString();
                    const value = data.value === 1 ? 'Không có xe' : 'có
xe';

                    // Kiểm tra nếu trạng thái hiện tại của cảm biến khác
giá trị mới nhận được
                    if (currentStatus !== value) {
                        // Cập nhật trạng thái hiện tại
                        currentStatus = value;

                        // Thêm dữ liệu mới vào mảng sensorData
                        sensorData.push({ label: label, value: value });

                        // Cập nhật bảng dữ liệu cảm biến
                        sensorDataContainer.innerHTML = '';
                        for (let i = 0; i < sensorData.length; i++) {
                            const entry = sensorData[i];
                            sensorDataContainer.innerHTML +=
`<tr><td>${entry.label}</td><td>${entry.value}</td></tr>`;
                        }

                        // Cập nhật biểu đồ
                        chart.data.labels = sensorData.map(entry =>
entry.label);
                        chart.data.datasets[0].data = sensorData.map(entry
=> entry.value === 'có xe' ? 1 : 0);
                        chart.update();
                    }
                });
            </script>




            <script>
                // Lấy tham chiếu đến phần tử tbody của SR505
                const sr505DataContainer =
document.getElementById('sr505_data');

                // Mảng lưu trữ dữ liệu của SR505
                const sr505Data = [];

                // Khởi tạo biểu đồ SR505
                const ctx1 =
document.getElementById('chart1').getContext('2d');
                const chart1 = new Chart(ctx1, {
                type: 'line',
                data: {
                    labels: [],
                    datasets: [{
                        label: 'Giá trị cảm biến hồng ngoại',
                        data: [],
                        backgroundColor: 'rgba(255, 255, 255, 0.2)', //
Màu nền trắng

                        borderColor: 'rgba(255, 255, 0, 1)', // Màu vàng
                        borderWidth: 5,
                        fill: true,
                    }]
                },
                options: {
                    scales: {
                        x: {
                            display: true,
                            grid: {
                                color: 'white' // Màu cột x trắng
                            },
                            ticks: {
                                color: 'darkgreen', // Màu thước đo y xanh
đậm

                                font: {
                                    size: 20, // Kích thước chữ to hơn
(14px)

                                    weight: 'bold' // Đậm
                                }
                            }
                        },

                        y: {
                            display: true,
                            suggestedMin: 0,
                            suggestedMax: 1,
                            stepSize: 1,
                            grid: {
                                color: 'blue' // Màu cột y trắng
                            },
                            ticks: {
                                color: 'darkgreen', // Màu thước đo y xanh
đậm

                                font: {
                                    size: 20, // Kích thước chữ to hơn
(14px)

                                    weight: 'bold' // Đậm
                                }
                            }
                        }
                    },
                    plugins: {
                        legend: {
                            labels: {
                                color: 'Black' // Màu chữ trắng
                            }
                        }
                    }
                }
            });

                // Lưu trữ trạng thái hiện tại của cảm biến SR505
                let currentStatusSR505 = '';

                // Lắng nghe sự kiện SSE và cập nhật dữ liệu cảm biến SR505
                const sr505EventSource = new EventSource('/sr505_data');
                sr505EventSource.addEventListener('message',
function(event) {
                    const data = JSON.parse(event.data);
                    const label = new Date(parseFloat(data.label) *
1000).toLocaleTimeString();
                    const value = data.value === 1 ? 'Không có người' :
'Có người trước cửa';

                    // Kiểm tra nếu trạng thái hiện tại của SR505 khác giá
trị mới nhận được
                    if (currentStatusSR505 !== value) {
                        // Cập nhật trạng thái hiện tại
                        currentStatusSR505 = value;

                        // Thêm dữ liệu mới vào mảng sr505Data
                        sr505Data.push({ label: label, value: value });

                        // Cập nhật bảng dữ liệu của SR505
                        sr505DataContainer.innerHTML = '';
                        for (let i = 0; i < sr505Data.length; i++) {
                            const entry = sr505Data[i];
                            sr505DataContainer.innerHTML +=
`<tr><td>${entry.label}</td><td>${entry.value}</td></tr>`;
                        }

                        // Cập nhật biểu đồ SR505
                        chart1.data.labels = sr505Data.map(entry =>
entry.label);
                        chart1.data.datasets[0].data = sr505Data.map(entry
=> entry.value === 'Có người trước cửa' ? 1 : 0);
                        chart1.update();
                    }
                });
            </script>
```

```
            <!-- Bảng thông số cảm biến DHT11 -->
            <div class="chart-wrapper1">
                <div class="chart-container2">
                    <canvas id="chart2"></canvas>
                </div>
            </div>

            <div style="overflow-y: auto; max-height: 200px; margin-left: 
auto; margin-right: auto; width: 700px;">
                <p class="highlight" style="font-size: 18px; padding-
bottom: 10px;">Bảng thông số cảm biến NHIỆT ĐỘ</p>
                <table style="font-size: 20px;">
                    <tr>
                        <th style="padding: 10px;">Thời gian</th>
                        <th style="padding: 10px;">Độ ẩm (g/m³)</th>
                        <th style="padding: 10px;">Nhiệt độ (°C)</th>
                    </tr>
                    <tbody id="dht11_data"></tbody>
                </table>
            </div>

            <script>
                // Lấy tham chiếu đến phần tử tbody
                const sensorDataContainerDHT = 
document.getElementById('dht11_data');

                // Mảng lưu trữ dữ liệu cảm biến
                const dht11data = [];

                // Khởi tạo biểu đồ
                const ctx2 = 
document.getElementById('chart2').getContext('2d');
                const chart2 = new Chart(ctx2, {
                    type: 'line',
                    data: {
                        labels: [],
                        datasets: [{
                            label: 'Nhiệt độ',
                            data: [],
                            backgroundColor: 'rgba(255, 255, 255, 0.2)',
                            borderColor: 'rgba(255, 0, 0, 1)',
                            borderWidth: 5,
                            fill: true,
                        }, {
                            label: 'Độ ẩm',
                            data: [],
                            backgroundColor: 'rgba(255, 255, 255, 0.2)',
                            borderColor: 'rgba(0, 0, 255, 1)',
                            borderWidth: 5,
                            fill: true,
                        }]
                    },
                    options: {
                        scales: {
                            x: {
                                display: true,
                                grid: {
                                    color: 'blue'
                                },
                                ticks: {
                                    color: 'darkgreen',
                                    font: {
                                        size: 20,
                                        weight: 'bold'
                                    }
                                }
                            },
                            y: {
                                display: true,
                                suggestedMin: 0,
                                suggestedMax: 100,
                                stepSize: 10,
                                grid: {
                                    color: 'blue'
                                },
                                ticks: {
                                    color: 'darkgreen',
                                    font: {
                                        size: 20,
                                        weight: 'bold'
                                    }
                                }
                            }
                        },
                        plugins: {
                            legend: {
                                labels: {
                                    color: 'black'
                                }
                            }
                        }
                    }
                });

                // Lưu trữ trạng thái hiện tại của cảm biến DHT11
                let currentStatusDHT11 = '';

                // Lắng nghe sự kiện SSE và cập nhật dữ liệu cảm biến DHT11
                const dht11EventSource = new EventSource('/dht11_data');
                dht11EventSource.addEventListener('message',
function(event) {
                    const data = JSON.parse(event.data);
                    const label = new Date(parseFloat(data.label) *
1000).toLocaleTimeString();
                    const humidity = data.humidity;
                    const temperature = data.temperature;
                    // Kiểm tra nếu trạng thái hiện tại của DHT11 khác giá
trị mới nhận được
                    if (currentStatusDHT11 !== label) {
                        // Cập nhật trạng thái hiện tại
                        currentStatusDHT11 = label;

                        // Thêm dữ liệu mới vào mảng sensorData
                        dht11data.push({ label: label, humidity: humidity,
temperature: temperature });

                        // Cập nhật bảng dữ liệu của sensorData
                        sensorDataContainerDHT.innerHTML = '';
                        for (let i = 0; i < dht11data.length; i++) {
                            const entry = dht11data[i];
                            sensorDataContainerDHT.innerHTML +=
`<tr><td>${entry.label}</td><td>${entry.humidity}</td><td>${entry.temperat
ure}</td></tr>`;
                        }

                        // Cập nhật biểu đồ sensorData
                        chart2.data.labels = dht11data.map(entry =>
entry.label);
                        chart2.data.datasets[1].data = dht11data.map(entry
=> entry.humidity);
                        chart2.data.datasets[0].data = dht11data.map(entry
=> entry.temperature);
                        chart2.update();
                    }
                });
            </script>
    </div>

</body>
</html>
```

4. Results

The implemented Flask application has successfully provided a user interface for controlling a wide range of devices and sensors connected to the Raspberry Pi. Users can interact with the application through the defined routes to perform various actions and access real-time sensor data.

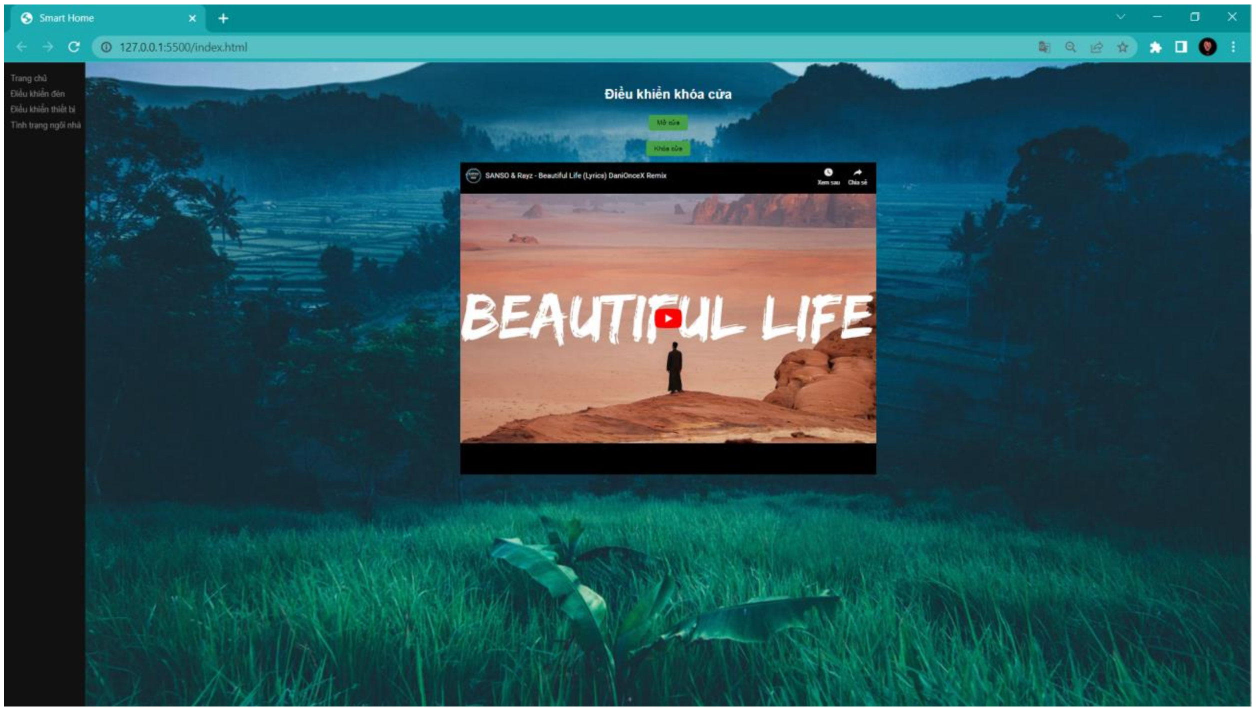
One of the key functionalities of the application is the ability to unlock and lock the door. By submitting the correct password through the designated route, users can unlock the door and grant access. Similarly, they can lock the door using the appropriate route. This feature enhances the security and convenience of the system.

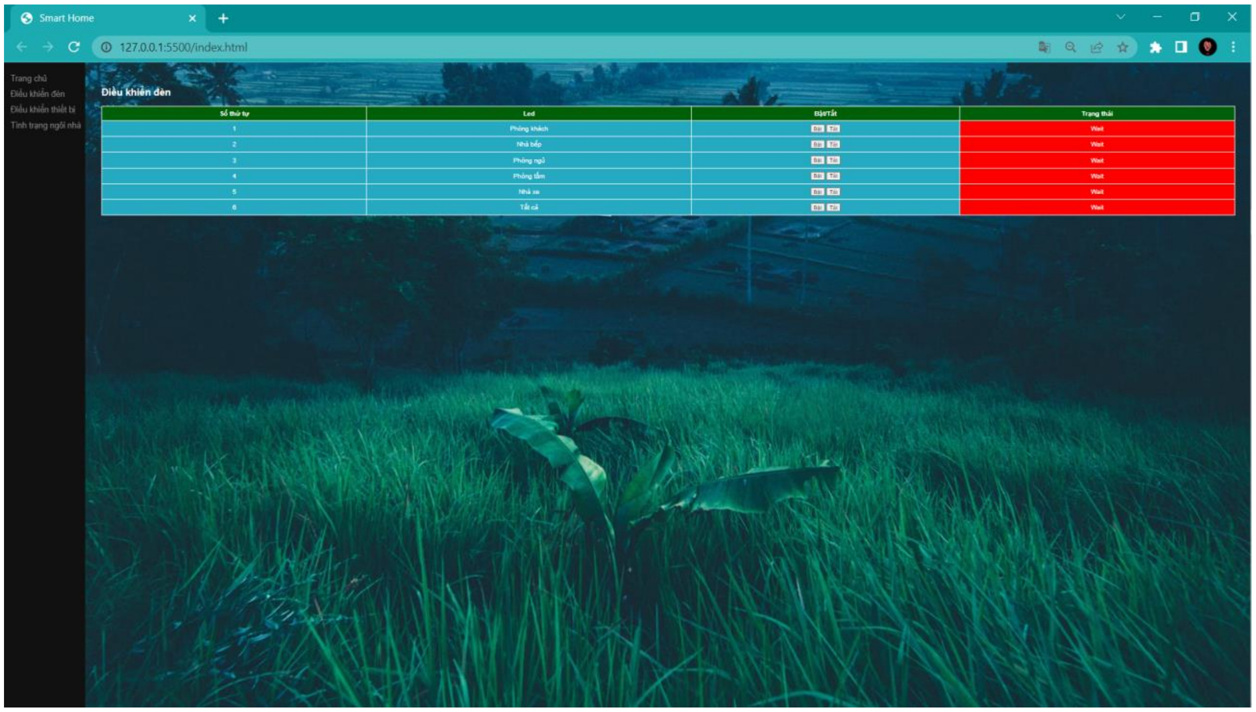
The application also allows users to control the lights in different rooms. Through the specified routes, users can turn the lights on or off in the kitchen, bedroom, bathroom, living room, and even control an additional LED light. This capability enables users to create desired lighting atmospheres and conserve energy by easily managing the illumination of different areas.

Furthermore, users can control the fan and TV using the dedicated routes. By sending commands through the application, they can turn the fan and TV on or off as per their requirements. This feature enhances the comfort and convenience of the environment, allowing users to adjust the room temperature and entertainment system effortlessly.

In addition to device control, the application provides real-time sensor data to users. By accessing the appropriate routes, users can view live readings from the SR505 motion sensor, SR505 sensor, and DHT11 temperature and humidity sensor. This functionality enables users to monitor the status and environmental conditions in real-time, promoting effective decision-making and ensuring optimal system performance.

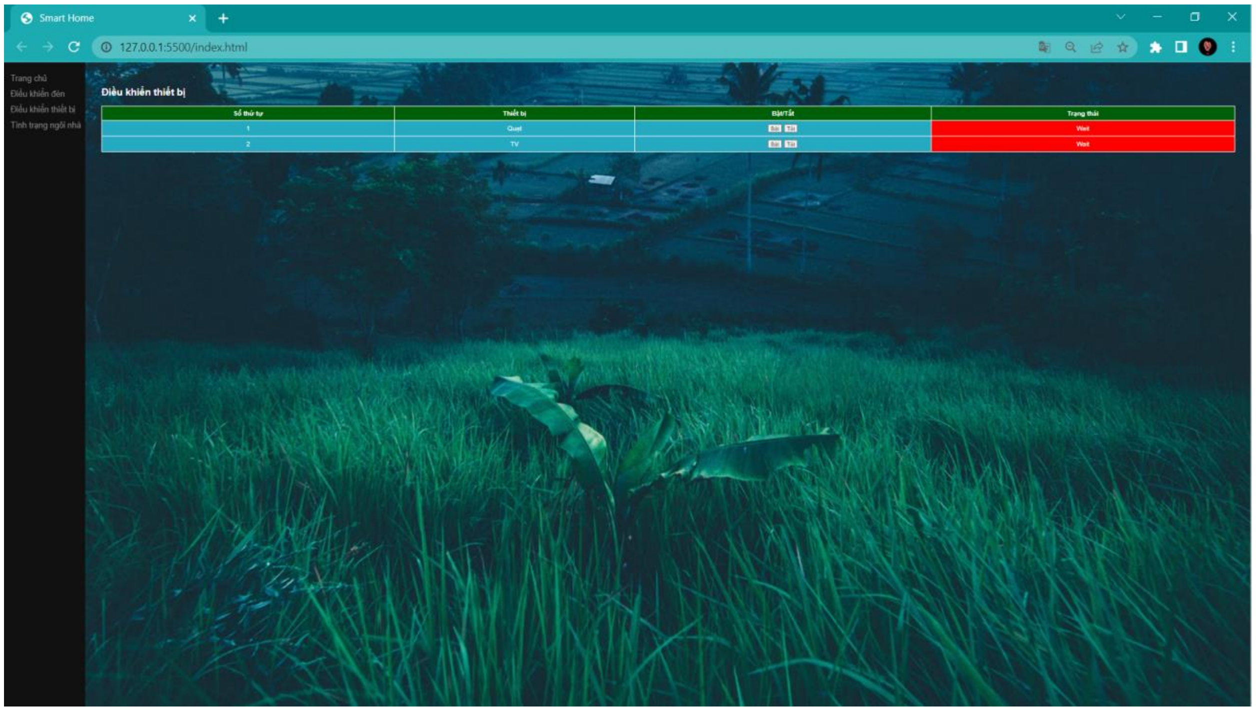
Overall, the implementation of the Flask application has yielded positive results in terms of device control and sensor monitoring. Users can conveniently unlock and lock the door, control lights, fan, and TV, and access real-time sensor data through the intuitive user interface. The application ensures the smooth functioning of the connected devices and provides accurate sensor readings, enhancing both security and convenience in the controlled environment.





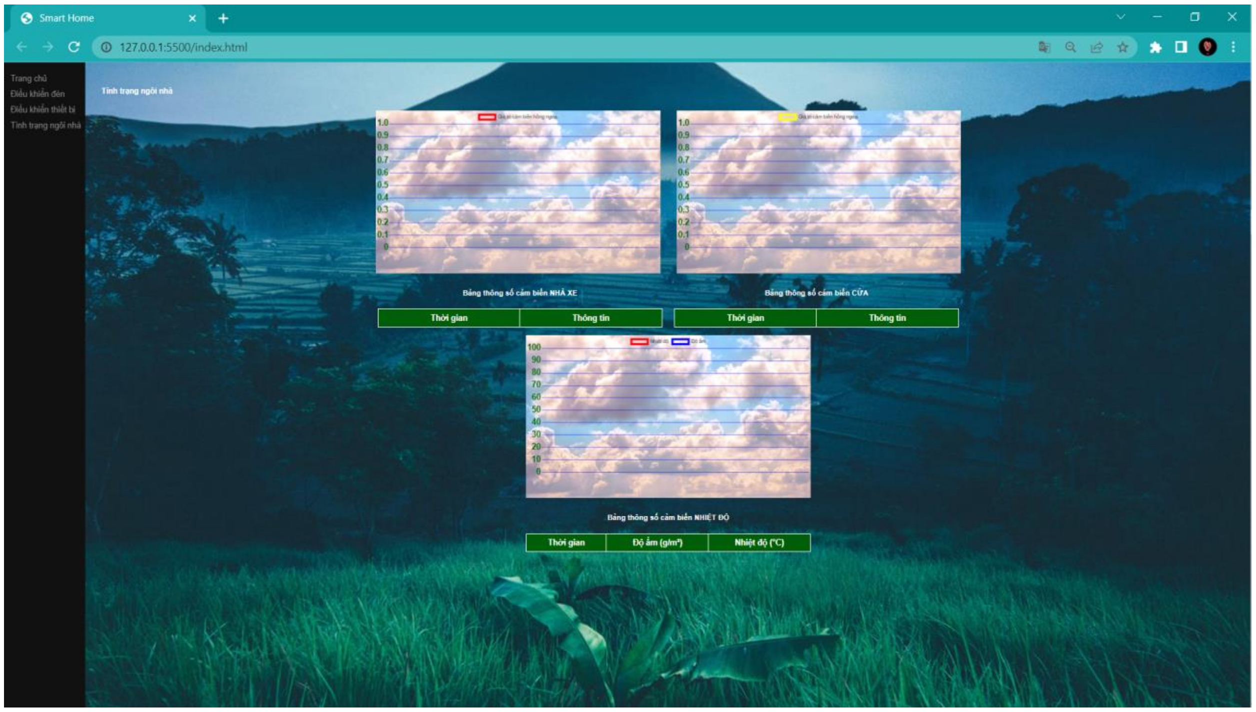
Điều khiển đèn

Số thứ tự	Led	Bật/Tắt	Trạng thái
1	Phòng khách	Bật Tắt	Wait
2	Nhà bếp	Bật Tắt	Wait
3	Phòng ngủ	Bật Tắt	Wait
4	Phòng tắm	Bật Tắt	Wait
5	Nhà xe	Bật Tắt	Wait
6	Tất cả	Bật Tắt	Wait



Điều khiển thiết bị

Số thứ tự	Thiết bị	Bật/Tắt	Trạng thái
1	Quạt	Bật Tắt	Wait
2	TV	Bật Tắt	Wait



Bảng thông số cảm biến NHÀ XE

Thời gian	Thông tin

Bảng thông số cảm biến CỬA

Thời gian	Thông tin

Bảng thông số cảm biến NHIỆT ĐỘ

Thời gian	Độ ẩm (g/m³)	Nhiệt độ (°C)

5. Conclusions

In conclusion, the Flask application developed for controlling devices and sensors connected to the Raspberry Pi has proven to be an effective and user-friendly solution. The application's implementation successfully enables users to interact with the system and perform various actions, such as unlocking and locking the door, controlling lights in different rooms, managing the fan and TV, and accessing real-time sensor data.

Through the intuitive user interface provided by the Flask routes, users can easily navigate and control the connected devices. The application's functionality enhances both security and convenience by allowing users to securely unlock and lock the door using a password and conveniently control the lights, fan, and TV based on their preferences.

Moreover, the implementation ensures the accurate and timely monitoring of sensor data. Users can access real-time readings from the SR505 motion sensor, SR505 sensor, and DHT11 temperature and humidity sensor. This feature enables users to make informed decisions, respond to environmental changes promptly, and maintain optimal conditions in the controlled environment.

Overall, the Flask application's results demonstrate its effectiveness in providing efficient device control and reliable sensor monitoring capabilities. The application's user interface and functionality enhance the user experience, promoting ease of use and convenience. With its successful implementation, the application proves to be a valuable tool for managing and controlling various devices and sensors connected to the Raspberry Pi.

REFERENCES

Flask: Official Flask Documentation

Adafruit_DHT: Adafruit_DHT Library

RPi.GPIO: RPi.GPIO Library