**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN:**

**ĐỀ TÀI:**

**Giáo viên hướng dẫn: ThS.**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã sv** | **Họ và tên** | **Lớp** |
|  |  |  |  |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
|  |  |  |  |  |  |

**CÁN BỘ CHẤM THI**

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

# MỤC LỤC

# MỤC LỤC HÌNH ẢNH

# KẾT LUẬN

# CHƯƠNG 1. GIỚI THIỆU

## 1.1. Mục tiêu của chương trình

Chương trình được phát triển với mục tiêu xây dựng một hệ thống Remote Desktop đơn giản nhưng hiệu quả, cho phép người dùng thực hiện các thao tác điều khiển máy tính từ xa thông qua một giao diện đồ họa trực quan và dễ sử dụng. Cụ thể, hệ thống hướng đến việc cung cấp khả năng thực hiện các tác vụ quan trọng như tắt máy, khởi động lại, mở ứng dụng hoặc chụp ảnh màn hình từ một máy tính khác trong cùng mạng nội bộ. Một trong những mục tiêu chính là hỗ trợ người dùng lập lịch cho các tác vụ này, giúp tự động hóa các hoạt động quản lý máy tính mà không cần can thiệp thủ công liên tục. Ví dụ, người dùng có thể đặt lịch để mở một ứng dụng cụ thể hoặc tắt máy vào một thời điểm xác định, từ đó tiết kiệm thời gian và tối ưu hóa quy trình làm việc. Bên cạnh đó, chương trình cũng tập trung vào việc cung cấp giao diện đồ họa thân thiện, sử dụng thư viện tkinter của Python, nhằm mang lại trải nghiệm mượt mà cho người dùng, ngay cả những người không quen thuộc với các công cụ dòng lệnh. Một mục tiêu khác là đảm bảo tính ổn định và bảo mật cơ bản thông qua cơ chế kết nối ngược (reverse connection), giúp vượt qua các rào cản như tường lửa hoặc NAT trong mạng nội bộ. Hệ thống được thiết kế để tương thích với các nền tảng phổ biến như Windows và Linux, đặc biệt trong các tác vụ hệ thống như tắt máy hoặc khởi động lại. Cuối cùng, chương trình hướng đến việc trở thành một giải pháp nhẹ, dễ triển khai, phù hợp cho các cá nhân hoặc nhóm nhỏ cần quản lý máy tính từ xa trong các môi trường như văn phòng hoặc gia đình.

## 1.2. Tổng quan về hệ thống Remote Desktop

Hệ thống Remote Desktop được thiết kế dựa trên mô hình client-server, trong đó hai thành phần chính là server (máy tính mục tiêu) và client (máy tính điều khiển) phối hợp chặt chẽ để thực hiện các chức năng điều khiển từ xa. Server chịu trách nhiệm lắng nghe và xử lý các lệnh từ client, chẳng hạn như thực hiện tắt máy, khởi động lại, mở ứng dụng, hoặc gửi ảnh chụp màn hình. Trong khi đó, client cung cấp một giao diện đồ họa để người dùng nhập lệnh và nhận phản hồi từ server, bao gồm cả dữ liệu văn bản và hình ảnh màn hình. Điểm nổi bật của hệ thống là sử dụng cơ chế kết nối ngược, trong đó client mở một cổng lắng nghe (mặc định là cổng 8081) và server chủ động kết nối đến client. Cơ chế này giúp giải quyết các vấn đề liên quan đến tường lửa hoặc cấu hình mạng phức tạp, đảm bảo kết nối ổn định trong môi trường mạng nội bộ. Các tính năng chính của hệ thống bao gồm khả năng đặt lịch tắt máy, khởi động lại, mở ứng dụng, chụp và truyền ảnh màn hình, cũng như hủy bỏ các lịch trình đã đặt. Tất cả các thao tác này được thực hiện thông qua giao diện đồ họa được xây dựng bằng thư viện tkinter, mang lại sự tiện lợi và dễ tiếp cận cho người dùng. Chương trình sử dụng Python làm ngôn ngữ chính, kết hợp với các thư viện như socket để xử lý giao tiếp mạng, apscheduler để lập lịch tác vụ, và mss cùng PIL để xử lý ảnh màn hình. Hệ thống phù hợp cho các ứng dụng trong mạng nội bộ, chẳng hạn như quản lý máy tính trong văn phòng, phòng thí nghiệm, hoặc gia đình, nơi người dùng cần thực hiện các tác vụ từ xa mà không cần trực tiếp ngồi trước máy tính mục tiêu.

## 1.3. Các thành phần chính của chương trình

Hệ thống Remote Desktop được xây dựng theo mô hình mô-đun, với các thành phần chính được tổ chức thành các tệp mã nguồn riêng biệt, mỗi tệp đảm nhận một vai trò cụ thể để đảm bảo tính rõ ràng và dễ bảo trì.

Thành phần đầu tiên là Server Controller (server\_controller.py), đóng vai trò là lõi logic của server. Tệp này chịu trách nhiệm thiết lập kết nối ngược đến client, xử lý các lệnh như tắt máy, khởi động lại, mở ứng dụng, hoặc chụp ảnh màn hình, đồng thời sử dụng thư viện apscheduler để lập lịch các tác vụ theo thời gian được chỉ định. Nó cũng sử dụng các thư viện như mss và PIL để chụp và nén ảnh màn hình trước khi gửi đến client, cùng với socket để giao tiếp mạng và threading để xử lý đa luồng.

Thành phần thứ hai là Client Controller (client\_controller.py), quản lý phía client trong việc thiết lập cổng lắng nghe cho kết nối ngược, gửi lệnh đến server và nhận phản hồi, bao gồm cả dữ liệu văn bản và ảnh màn hình. Tệp này cũng hỗ trợ lấy địa chỉ IP cục bộ của client để gửi đến server, đảm bảo kết nối chính xác.

Thành phần thứ ba là Giao diện người dùng Server (server\_gui.py), cung cấp giao diện đồ họa cho server, cho phép người dùng nhập địa chỉ IP của client và chọn các tác vụ thông qua các nút chức năng như bắt đầu server, đặt lịch tắt máy, hoặc hủy lịch trình. Giao diện này sử dụng tkinter và hiển thị một logo được điều chỉnh kích thước để phù hợp với bố cục. Tương tự, Giao diện người dùng Client (client\_gui.py) cung cấp giao diện cho client, hiển thị địa chỉ IP cục bộ và các tùy chọn điều khiển thông qua menu chính, đồng thời tích hợp khả năng sao chép IP vào bộ nhớ tạm bằng pyperclip. Các trang giao diện khác, bao gồm shutdown\_page.py, restart\_page.py, open\_app\_page.py, và cancel\_schedule\_page.py, cung cấp các form nhập liệu để người dùng đặt lịch hoặc hủy lịch trình, với mỗi trang được thiết kế để hỗ trợ một tác vụ cụ thể như nhập giờ/phút hoặc tên ứng dụng.

Các thành phần phụ trợ bao gồm tệp logo (img/logo.png) và các thư viện như socket, mss, và PIL, hỗ trợ giao tiếp mạng và xử lý ảnh. Sự phân chia rõ ràng này giúp hệ thống dễ dàng mở rộng và bảo trì, đồng thời đảm bảo các chức năng được thực hiện một cách hiệu quả và đáng tin cậy.

# CHƯƠNG 2. THIẾT KẾ VÀ KIẾN TRÚC

## 2.1. Cấu trúc tổng thể của hệ thống

Hệ thống Remote Desktop được thiết kế theo mô hình client-server, trong đó hai thành phần chính là máy chủ (server) và máy khách (client) phối hợp chặt chẽ để thực hiện các chức năng điều khiển từ xa. Cấu trúc tổng thể của hệ thống được xây dựng dựa trên nguyên tắc mô-đun, với các thành phần riêng biệt chịu trách nhiệm cho các nhiệm vụ cụ thể như giao tiếp mạng, xử lý lệnh, lập lịch tác vụ, và cung cấp giao diện đồ họa. Mỗi thành phần được tổ chức trong các tệp mã nguồn riêng, giúp hệ thống dễ dàng bảo trì, mở rộng và kiểm tra.

Hệ thống sử dụng cơ chế kết nối ngược (reverse connection) để thiết lập liên kết giữa client và server. Trong mô hình này, client khởi tạo một máy chủ lắng nghe (listener) trên cổng 8081, và server chủ động kết nối đến client. Cơ chế này giúp vượt qua các hạn chế của tường lửa hoặc NAT trong mạng nội bộ, đảm bảo kết nối đáng tin cậy mà không cần cấu hình mạng phức tạp. Giao tiếp giữa client và server được thực hiện thông qua giao thức TCP, sử dụng thư viện socket của Python để đảm bảo truyền dữ liệu ổn định và an toàn ở mức cơ bản. Các lệnh được gửi dưới dạng chuỗi ký tự (text) hoặc dữ liệu nhị phân (cho ảnh chụp màn hình), với định dạng cụ thể để đảm bảo server và client hiểu rõ yêu cầu và phản hồi.

Về mặt chức năng, hệ thống cung cấp các khả năng chính như sau: đặt lịch tắt máy, khởi động lại, mở ứng dụng, chụp và truyền ảnh màn hình, cũng như hủy tất cả lịch trình đã đặt. Các tác vụ này được thực hiện thông qua giao diện đồ họa (GUI) được xây dựng bằng thư viện tkinter, mang lại trải nghiệm trực quan cho người dùng. Server sử dụng apscheduler để quản lý lịch trình, cho phép thực thi các tác vụ tại thời điểm được chỉ định. Ảnh chụp màn hình được xử lý bằng mss để chụp và PIL để nén trước khi gửi qua mạng. Tất cả các thành phần giao diện và logic được tích hợp chặt chẽ, với luồng dữ liệu được thiết kế để đảm bảo hiệu suất và tính đồng bộ.

Cấu trúc tổng thể của hệ thống có thể được mô tả qua các tầng chính:

* Tầng giao diện người dùng: Bao gồm các giao diện đồ họa (GUI) cho cả client và server, được triển khai trong các tệp như server\_gui.py, client\_gui.py, và các trang giao diện (shutdown\_page.py, restart\_page.py, open\_app\_page.py, cancel\_schedule\_page.py). Tầng này chịu trách nhiệm thu thập đầu vào từ người dùng và hiển thị phản hồi từ hệ thống.
* Tầng điều khiển (controller): Bao gồm server\_controller.py và client\_controller.py, xử lý logic cốt lõi như giao tiếp mạng, thực thi lệnh, và quản lý lịch trình.
* Tầng giao tiếp mạng: Sử dụng thư viện socket để thiết lập kết nối TCP và truyền dữ liệu (lệnh, phản hồi, hoặc ảnh chụp màn hình) giữa client và server.
* Tầng xử lý dữ liệu: Bao gồm các thư viện như mss, PIL để xử lý ảnh chụp màn hình, và apscheduler để quản lý lịch trình tác vụ.

A diagram of a remote desktop

Description automatically generated

Hình 1. Cấu trúc tổng thể của hệ thống

Hệ thống được thiết kế để hoạt động trong mạng nội bộ, phù hợp cho các ứng dụng như quản lý máy tính trong văn phòng, phòng thí nghiệm, hoặc gia đình. Sự phân chia mô-đun rõ ràng và việc sử dụng các thư viện Python tiêu chuẩn giúp hệ thống dễ triển khai và mở rộng trong tương lai.

## 2.2. Mô tả các module chính

Hệ thống Remote Desktop được chia thành các module chính, mỗi module đảm nhận một vai trò cụ thể trong việc thực hiện các chức năng của hệ thống. Các module này được tổ chức trong các tệp mã nguồn riêng biệt, với sự tương tác chặt chẽ để đảm bảo hoạt động mượt mà và hiệu quả. Dưới đây là mô tả chi tiết về từng module chính.

### 2.2.1. Server Controller

A diagram of a server controller

Description automatically generated

Module Server Controller (server\_controller.py) là lõi logic của phía server, chịu trách nhiệm xử lý các yêu cầu từ client và thực thi các tác vụ trên máy tính mục tiêu. Module này sử dụng một lớp ServerController để quản lý tất cả các hoạt động của server, bao gồm khởi tạo kết nối, xử lý lệnh, lập lịch tác vụ, và truyền dữ liệu. Các chức năng chính của module bao gồm:

* Khởi tạo và quản lý kết nối: Server tạo một socket TCP và lắng nghe kết nối từ client trên cổng 8080. Khi nhận được địa chỉ IP từ client, server sử dụng cơ chế kết nối ngược để kết nối đến client trên cổng 8081. Điều này được thực hiện thông qua phương thức reverse\_connect, giúp đảm bảo kết nối ổn định ngay cả trong môi trường có tường lửa hoặc NAT.
* Xử lý lệnh từ client: Module nhận các lệnh từ client thông qua phương thức handle\_client, được chạy trong một luồng riêng để xử lý đồng thời nhiều yêu cầu. Các lệnh được hỗ trợ bao gồm:
  + shutdown: Lên lịch tắt máy tại thời điểm xác định, sử dụng apscheduler để thêm công việc vào lịch trình.
  + restart: Lên lịch khởi động lại máy, tương tự như lệnh shutdown.
  + open\_app: Lên lịch mở ứng dụng (mặc định hỗ trợ Chrome, hoặc các ứng dụng khác thông qua os.startfile).
  + cancel\_schedule: Hủy tất cả các lịch trình đã đặt.
  + start\_stream và stop\_stream: Bắt đầu hoặc dừng truyền ảnh chụp màn hình liên tục.
  + screenshot: Chụp và gửi một ảnh màn hình duy nhất.
  + upload\_file: Nhận và lưu tệp từ client.
* Chụp và truyền ảnh màn hình: Module sử dụng thư viện mss để chụp ảnh màn hình và PIL để nén ảnh dưới định dạng JPEG trước khi gửi qua socket. Phương thức send\_screenshot xử lý việc chụp, nén và truyền ảnh, đảm bảo kích thước dữ liệu được tối ưu hóa để giảm tải mạng.
* Lập lịch tác vụ: Module tích hợp apscheduler để quản lý các tác vụ được lên lịch, chẳng hạn như tắt máy, khởi động lại hoặc mở ứng dụng. Các phương thức như schedule\_shutdown, schedule\_restart, và schedule\_open\_app tính toán thời điểm thực thi dựa trên đầu vào từ client (giờ và phút) và thêm công việc vào lịch trình.
* Hỗ trợ đa nền tảng: Module xử lý các lệnh hệ thống như tắt máy hoặc khởi động lại khác nhau tùy thuộc vào hệ điều hành (Windows hoặc Linux), sử dụng các lệnh như shutdown /s /t 0 (Windows) hoặc shutdown -h now (Linux).z

Module sử dụng các thư viện như socket, os, pyautogui, mss, PIL, apscheduler, datetime, và threading để đảm bảo hoạt động hiệu quả và đáng tin cậy. Cấu trúc của module được thiết kế để xử lý các ngoại lệ (exception) và đóng kết nối đúng cách khi xảy ra lỗi, đảm bảo tính ổn định của hệ thống.

### 2.2.2. Client Controller

Module Client Controller (client\_controller.py) chịu trách nhiệm quản lý phía client trong việc thiết lập kết nối mạng và giao tiếp với server. Module này triển khai lớp ClientController với các chức năng chính sau:

* Lấy địa chỉ IP cục bộ: Phương thức get\_local\_ip sử dụng một socket tạm thời để kết nối đến một địa chỉ công cộng (8.8.8.8) và lấy địa chỉ IP cục bộ của client. Địa chỉ này được gửi đến server để thiết lập kết nối ngược.
* Lắng nghe kết nối ngược: Phương thức start\_reverse\_listener tạo một socket TCP lắng nghe trên cổng 8081, chờ server kết nối đến. Khi kết nối được thiết lập, client lưu trữ đối tượng socket (reverse\_conn) để sử dụng cho giao tiếp tiếp theo.
* Gửi và nhận dữ liệu: Module cung cấp các phương thức receive\_data và receive\_image để nhận dữ liệu văn bản (phản hồi từ server) hoặc ảnh chụp màn hình. Dữ liệu được gửi và nhận dưới dạng nhị phân, với kích thước dữ liệu được gửi trước dưới dạng 4 byte để đảm bảo nhận chính xác.
* Đóng kết nối: Phương thức close đảm bảo đóng cả socket lắng nghe và kết nối ngược khi không còn cần thiết, giúp giải phóng tài nguyên.

Module này sử dụng thư viện socket và PIL để xử lý giao tiếp mạng và ảnh chụp màn hình. Thiết kế của module tập trung vào sự đơn giản và hiệu quả, với các phương thức được tối ưu hóa để đảm bảo giao tiếp đáng tin cậy giữa client và server.

### 2.2.3. Giao diện người dùng Server (server\_gui.py)

### 2.2.4. Giao diện người dùng Client (client\_gui.py)

### 2.2.5. Các trang giao diện (shutdown\_page.py, restart\_page.py, open\_app\_page.py, cancel\_schedule\_page.py)

## 2.3. Luồng dữ liệu và giao tiếp giữa client-server

## 2.4. Công nghệ và thư viện sử dụng

# CHƯƠNG 3. CHỨC NĂNG VÀ KẾT QUẢ

3.1. Các chức năng chính của chương trình

3.1.1. Kết nối ngược (Reverse Connection)

3.1.2. Điều khiển từ xa (Shutdown, Restart, Open App)

3.1.3. Hủy lịch trình

3.1.4. Chụp và truyền ảnh màn hình

3.2. Kết quả thực hiện

3.3. Hạn chế và định hướng phát triển

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] William Stallings, Cryptography and Network Security: Principles and Practice, 7th Edition, Pearson, 2017.

[2] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, Wiley, 1996.

[3] National Institute of Standards and Technology (NIST), Data Encryption Standard (TRIPLEDES), FIPS Publication 46-3, 1999. <https://csrc.nist.gov/publications/detail/fips/46/3/final>

[4] Rivest, R. L., Shamir, A., & Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.

[5] RFC 3447, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, Internet Engineering Task Force (IETF), 2003. https://www.rfc-editor.org/rfc/rfc3447

[6] Eastlake, D., & Jones, P., "US Secure Hash Algorithm 2 (SHA-256, SHA-384, SHA-512)," RFC 4634, IETF, 2006. https://www.rfc-editor.org/rfc/rfc4634

[7] Vue.js Official Documentation, <https://vuejs.org/>

[8] Node.js Official Documentation, <https://nodejs.org/>

[9] Socket.IO Official Documentation, <https://socket.io/>

[10] JSEncrypt: RSA Encryption in JavaScript, <https://github.com/travist/jsencrypt>

[11] Trần Văn Tùng, An toàn và bảo mật hệ thống mạng, NXB Bưu điện, 2020.

[12] Nguyễn Văn Minh, Ứng dụng mật mã trong bảo mật dữ liệu truyền thông, Luận văn Thạc sĩ, Trường Đại học Bách Khoa Hà Nội, 2021.