

**ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN**



**PBL4: DỰ ÁN HỆ ĐIỀU HÀNH
& MẠNG MÁY TÍNH**

Đề tài: Ứng dụng điều khiển máy tính từ xa

SINH VIÊN THỰC HIỆN:

Phan Thanh Tâm

Lớp: 21T_DT

Nhóm: 2

Lưu Văn Duy

Lớp: 21T_DT

Nhóm: 2

GIẢNG VIÊN HƯỚNG DẪN: TS. Nguyễn Công Danh

Đà Nẵng, 12/2023

MỤC LỤC

MỤC LỤC.....	2
DANH SÁCH HÌNH VẼ	4
DANH SÁCH CÁC TỪ VIẾT TẮT.....	5
GIỚI THIỆU ĐỀ TÀI	6
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT	7
1.1. Mô hình client – server.....	7
1.1.1. Mô tả	7
1.1.2. Các mô hình xử lý client-server:	7
1.2. Mô hình TCP/IP.....	7
1.2.1. Định nghĩa.....	7
1.2.2. Cách thức hoạt động	8
1.2.3. Các tầng trong mô hình TCP/IP	8
1.3. Kỹ thuật điều khiển máy tính từ xa	10
1.3.1. Mô tả	10
1.3.2. Quy trình kết nối	10
1.3.3. Cách thức thực hiện	10
1.3.4. Mô tả kĩ thuật	11
CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....	13
2.1. Yêu cầu hệ thống.....	13
2.2. Phân tích yêu cầu	13

PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH	
2.2.1. Yêu cầu chức năng.....	13
2.2.2. Yêu cầu phi chức năng	13
2.3. Mô tả chung về hệ thống	14
2.4. Tổ chức mã nguồn.....	15
2.4.1. Client :	15
2.4.2. Server :	15
2.4.3 Thư viện tự tạo:	15
2.5. Phân tích vận hành hệ thống	16
2.5.1 Truyền tín hiệu hình ảnh	16
2.5.2 Truyền tín hiệu điều khiển chuột.....	17
2.5.3 Truyền tín hiệu điều khiển bàn phím.....	18
2.5.4 Truyền tín hiệu âm thanh	20
CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ.....	22
3.1. Môi trường cài đặt:	22
3.2. Giao diện hệ thống.....	23
3.2.1. Giao diện chương trình server	23
3.2.2. Giao diện chương trình client.....	25
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	29
1. Kết luận:	29
2. Hướng phát triển:.....	29
TÀI LIỆU THAM KHẢO	30
PHỤ LỤC.....	31

DANH SÁCH HÌNH VẼ

Hình 1.1: Cấu trúc mô hình TCP/IP	8
Hình 2.1: Sơ đồ use-case của hệ thống	14
Hình 2.2: Đoạn mã thiết lập chức năng chụp ảnh màn hình.....	19
Hình 2.3: Bảng mã các thao tác chuột	19
Hình 2.4: Các hàm xử lý các thao tác chuột.....	19
Hình 2.5: Đoạn mã để Hook sự kiện bàn phím	19
Hình 2.6: Đoạn mã nhận tín hiệu âm thanh vào	19
Hình 2.7: Đoạn mã phát tín hiệu âm thanh.....	219
Hình 3.1: Giao diện chương trình server	23
Hình 3.2: Server đang lắng nghe kết nối và bật kết nối âm thanh.....	20
Hình 3.3: Giao diện chương trình client	21
Hình 3.3: Giao diện màn hình điều khiển.....	21
Hình 3.3: Thực hiện các thao tác điều khiển với bàn phím	21
Hình 3.3: Thực hiện thao tác điều khiển với chuột và mở kết nối âm thanh.....	21
Hình 3.3: Giao tiếp bằng chức năng "Chat"(giao diện Client)	21
Hình 3.3: Giao tiếp bằng chức năng "Chat"(giao diện Server)	21

DANH SÁCH CÁC TỪ VIẾT TẮT

Từ viết tắt	Diễn giải
TCP/IP	Transmission Control Protocol/Internet Protocol
API	Application Programming Interface
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
OSI	Open Systems Interconnection

GIỚI THIỆU ĐỀ TÀI

Với sự tiến triển của internet, sự ra đời của máy tính và đa dạng phần mềm, cũng như sự phổ biến ngày càng rộng rãi của máy tính và mạng internet, việc sử dụng máy tính và các công cụ liên quan đã trở thành một kỹ năng quan trọng trong cuộc sống hàng ngày. Tuy nhiên, đối với những người ít hoặc chưa tiếp xúc nhiều với máy tính, việc này có thể là một thách thức. Hơn nữa, với sự xuất hiện của nhiều phần mềm mới đòi hỏi cài đặt phức tạp, hoặc khi máy tính gặp phải các vấn đề kỹ thuật mà người dùng không biết cách xử lý, điều này có thể dẫn đến những tình huống khó khăn và gây ra lỗi nghiêm trọng. Trong những tình huống như vậy, không ít người mong muốn nhận được sự hỗ trợ từ những người có kiến thức sâu rộng về máy tính, những người có khả năng hướng dẫn và giúp đỡ họ trong việc thao tác trên máy tính một cách chính xác. Tuy nhiên, việc hỗ trợ trực tiếp có thể gặp nhiều khó khăn, đặc biệt là khi người dùng ở xa.

Nhận thức được những khó khăn này, nhóm chúng em đã tiến hành nghiên cứu và đưa ra giải pháp thông qua dự án "**Ứng dụng Điều khiển Máy Tính Từ Xa**". Ứng dụng này cho phép người dùng có thể điều khiển máy tính của người khác từ xa, giúp họ thực hiện các thao tác trên máy tính một cách dễ dàng và chính xác. Mục tiêu của dự án là cung cấp sự giúp đỡ cho những người chưa thành thạo về máy tính, cho phép họ nhận được sự hướng dẫn và hỗ trợ trực tiếp trên máy tính của mình từ người sử dụng máy tính có kinh nghiệm.

Hy vọng đây sẽ là giải pháp góp phần giúp người dùng máy tính mới hoặc ít kinh nghiệm sử dụng máy tính một cách thuận tiện và hiệu quả hơn.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Mô hình client – server

1.1.1. Mô tả

Là mô hình gồm 3 thành phần cơ bản: front-end client, back-end server và mạng máy tính (network).

Chương trình front-end client chạy trên máy người dùng, ở đó họ có thể giao tiếp với ứng dụng để yêu cầu cung cấp dịch vụ, ví dụ như truy vấn dữ liệu.

Chương trình back-end server chạy trên máy chủ, tiếp nhận thông tin và cung cấp dịch vụ được yêu cầu, ví dụ như phản hồi truy vấn.

Mạng máy tính có chức năng truyền tải thông tin giữa back-end server và front-end client.

1.1.2. Các mô hình xử lý client-server:

Có 4 mô hình xử lý phổ biến:

Mô hình trình bày từ xa: mô hình này server thực hiện truy cập dữ liệu, xử lý tính toán. Kết quả tính toán được trả về và trình bày trên client.

Mô hình xử lý phân tán: mô hình này chia sẻ năng lực tính toán trên server cho client. Client ngoài chức năng quản lý giao diện còn được phân bổ một số chức năng xử lý thích hợp.

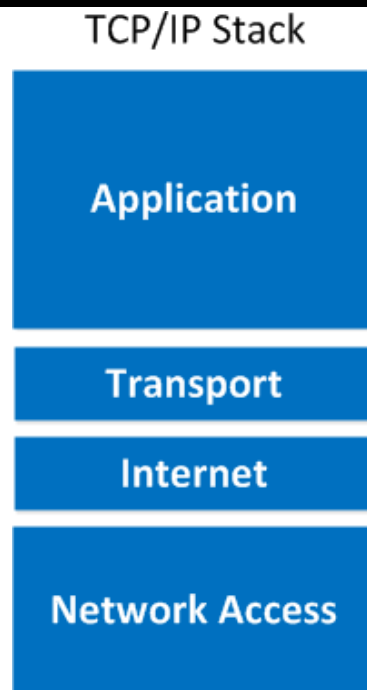
Mô hình quản lý dữ liệu từ xa: mô hình này tận dụng khả năng tính toán của client. Server chỉ có chức năng quản lý dữ liệu, còn client có trách nhiệm xử lý tính toán và quản lý giao diện người dùng.

Mô hình quản lý dữ liệu phân tán: toàn bộ tính toán tập trung trên client. Tuy nhiên một phần dữ liệu được quản lý bởi server.

1.2. Mô hình TCP/IP

1.2.1. Định nghĩa

TCP/IP (Transmission Control Protocol/Internet Protocol – giao thức điều khiển truyền nhận/giao thức liên mạng) là một bộ giao thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet.



Hình 1.1: Cấu trúc mô hình TCP/IP

1.2.2. Cách thức hoạt động

Giao thức IP cho phép các gói tin được gửi đến đích đã định sẵn, bằng cách thêm các thông tin dẫn đường (chính là Header) vào các gói tin để các gói tin đến đúng đích đã được định sẵn ban đầu. Giao thức TCP đóng vai trò kiểm tra và đảm bảo sự an toàn cho mỗi gói tin khi đi qua mỗi trạm. Trong quá trình này, nếu giao thức TCP nhận thấy gói tin bị lỗi, một tín hiệu sẽ được truyền đi và yêu cầu hệ thống gửi lại một gói tin khác.

Dữ liệu truyền vào mỗi tầng là khác nhau và được gọi tên như sau:

- Tầng Network Access: dữ liệu được gọi là frame
- Tầng Internet: dữ liệu được gọi là các IP Datagram
- Tầng Transport: dữ liệu được gọi là TCP segment
- Tầng Application: dữ liệu được gọi là stream

1.2.3. Các tầng trong mô hình TCP/IP

Gồm 4 tầng, bắt đầu từ tầng thấp nhất là Network Access, Internet, Transport và Application

Tầng 1: Network Access

Thực hiện chức năng giao tiếp môi trường mạng, chuyển giao dòng dữ liệu lên đường truyền vật lý.

Thực hiện chức năng tương đương lớp 1, 2 của mô hình OSI.

Tầng 2: Internet

Thực hiện các chức năng xử lý và truyền gói tin trên mạng. Các quá trình định tuyến sẽ được thực hiện ở lớp này.

Có các giao thức gồm IP, ICMP, IGMP.

Tầng 3: Transport

Thực hiện chức năng chuyển vận luồng dữ liệu giữa 2 trạm. Đảm bảo độ tin cậy, điều khiển luồng, phát hiện và sửa lỗi.

Có 2 giao thức chính là TCP và UDP.

Tầng 4: Application

Cung cấp các chương trình ứng dụng trên mạng TCP/IP.

Thực hiện các chức năng của các lớp cao nhất trong mô hình 7 lớp OSI bao gồm: mã hóa/giải mã, nén, định dạng dữ liệu, thiết lập/giải phóng phiên giao dịch.

1.3. Kỹ thuật điều khiển máy tính từ xa

1.3.1. Mô tả

Kỹ thuật điều khiển máy tính từ xa thông qua giao thức TCP hoạt động dựa trên nguyên tắc kết nối giữa hai máy tính, máy chủ và máy khách, thông qua một kênh truyền tin an toàn và ổn định. Máy chủ là máy tính mà người dùng muốn điều khiển, máy khách là máy tính mà người dùng sử dụng để điều khiển máy chủ.

1.3.2. Quy trình kết nối

Quy trình kết nối giữa máy chủ và máy khách được thực hiện như sau:

1. Máy khách gửi yêu cầu kết nối đến máy chủ.
2. Máy chủ chấp nhận hoặc từ chối yêu cầu kết nối.
3. Nếu máy chủ chấp nhận yêu cầu kết nối, máy chủ sẽ tạo một kênh truyền tin TCP cho máy khách.
4. Máy khách kết nối với kênh truyền tin TCP của máy chủ.
5. Máy chủ và máy khách có thể sử dụng loa và micro để kết nối âm thanh trực tiếp với nhau phục vụ quá trình giao tiếp, hoặc có thể chat thông qua khung chat.

1.3.3. Cách thức thực hiện

1. Phần mềm điều khiển được cài đặt trên cả máy tính cần hỗ trợ và máy tính điều khiển. Phần mềm này cho phép máy tính thiết lập kết nối TCP và UDP với nhau.
2. Khi người dùng máy tính cần hỗ trợ kích hoạt tính năng điều khiển, phần mềm sẽ tạo ra một port nghe để chấp nhận kết nối từ máy tính điều khiển.
3. Người dùng máy tính điều khiển cũng kích hoạt tính năng này và nhập địa chỉ IP của máy tính cần hỗ trợ. Phần mềm sẽ thiết lập kết nối TCP đến port đó.
4. Sau khi kết nối thành công, dữ liệu điều khiển như màn hình, click chuột, nhấn phím sẽ được gửi qua kênh kết nối TCP này từ máy tính điều khiển xuống máy tính cần hỗ trợ.
5. Máy tính được hỗ trợ sẽ nhận dữ liệu điều khiển này và thực hiện các hành động như click chuột, nhập liệu phím tương ứng.
6. Người dùng máy tính cần hỗ trợ có thể theo dõi và tương tác với máy tính điều khiển qua giao diện phần mềm đồng thời, giao tiếp với người dùng khác thông qua tính năng chat hoặc gọi âm thanh.

1.3.4. Mô tả kỹ thuật

1. Thiết lập Kết nối TCP:

- Quá trình bắt đầu bằng việc thiết lập kết nối TCP giữa máy tính điều khiển (client) và máy tính được điều khiển (server). Điều này thường sử dụng một cặp địa chỉ IP và cổng để xác định máy chủ.

2. Xác thực và Ổn định Kết nối:

- Sau khi kết nối được thiết lập, quá trình xác thực có thể được thực hiện để đảm bảo tính bảo mật của kết nối. Các phương thức xác thực địa chỉ host và mật khẩu. Sau khi xác thực thành công thì đến bước tiếp theo, nếu không chương trình sẽ trả về lỗi xác thực.

3. Gửi và Nhận Dữ liệu:

- Khi kết nối được xác thực, dữ liệu điều khiển có thể được gửi giữa máy tính điều khiển và máy tính được điều khiển. Điều này bao gồm các tương tác như chia sẻ màn hình, bấm phím, thao tác chuột và truyền tải các lệnh từ máy tính điều khiển đến máy tính được điều khiển.
- Chia sẻ màn hình (TCP)
- Thao tác chuột (TCP)
- Thao tác bàn phím (TCP)
- Kết nối âm thanh (UDP)
- Chat (TCP)

4. Quy trình Đồng bộ Hóa:

- Cơ chế đồng bộ hóa đảm bảo rằng các tương tác diễn ra một cách đồng bộ. Chẳng hạn, khi người dùng nhấn một phím, thông điệp tương ứng được gửi đến máy tính được điều khiển, và máy tính này phản hồi kết quả (nếu có).

5. Quản lý Dữ liệu Đa hướng:

- Điều khiển từ xa thường bao gồm truyền tải dữ liệu đa hướng, chẳng hạn như âm thanh và hình ảnh màn hình. Điều này yêu cầu quản lý dữ liệu đa hướng một cách hiệu quả để đảm bảo trải nghiệm người dùng không bị gián đoạn.

6. Bảo mật và Mã hóa:

- Việc sử dụng mã hóa là quan trọng để bảo vệ thông tin truyền tải qua mạng, đặc biệt là khi có các tương tác nhạy cảm như nhập mật khẩu hoặc truy cập dữ liệu quan trọng.

7. Quản lý Phiên:

- Hệ thống cần có khả năng quản lý phiên để theo dõi trạng thái kết nối và đảm bảo rằng phiên làm việc giữa máy tính điều khiển và máy tính được điều khiển có thể duy trì một cách ổn định.

8. Xử lý Lỗi và Khôi phục:

- Cơ chế xử lý lỗi cần được tích hợp để giải quyết các vấn đề kết nối, đảm bảo tính ổn định và liên tục của dịch vụ.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1. Yêu cầu hệ thống

- Giao diện đơn giản, thân thiện với người dùng, dễ dàng thao tác sử dụng.
- Sử dụng trên nền tảng ứng dụng máy tính window.
- Cho phép điều khiển được máy tính khác từ xa thông qua địa chỉ IP, dữ liệu được truyền nhanh chóng, chính xác, độ phân giải màn hình ổn định.
- Có thể cung cấp thêm một số chức năng nhắn tin hoặc gọi thoại cơ bản phục vụ mục đích giao tiếp.

2.2. Phân tích yêu cầu

2.2.1. Yêu cầu chức năng

- Sử dụng giao thức TCP kết hợp UDP để truyền giữ liệu.
- Sử dụng mô hình client-server.
- Cho phép người dùng kết nối chính xác, đảm bảo tính bảo mật.
- Truyền dữ liệu ổn định, nhanh chóng, chính xác.

2.2.2. Yêu cầu phi chức năng

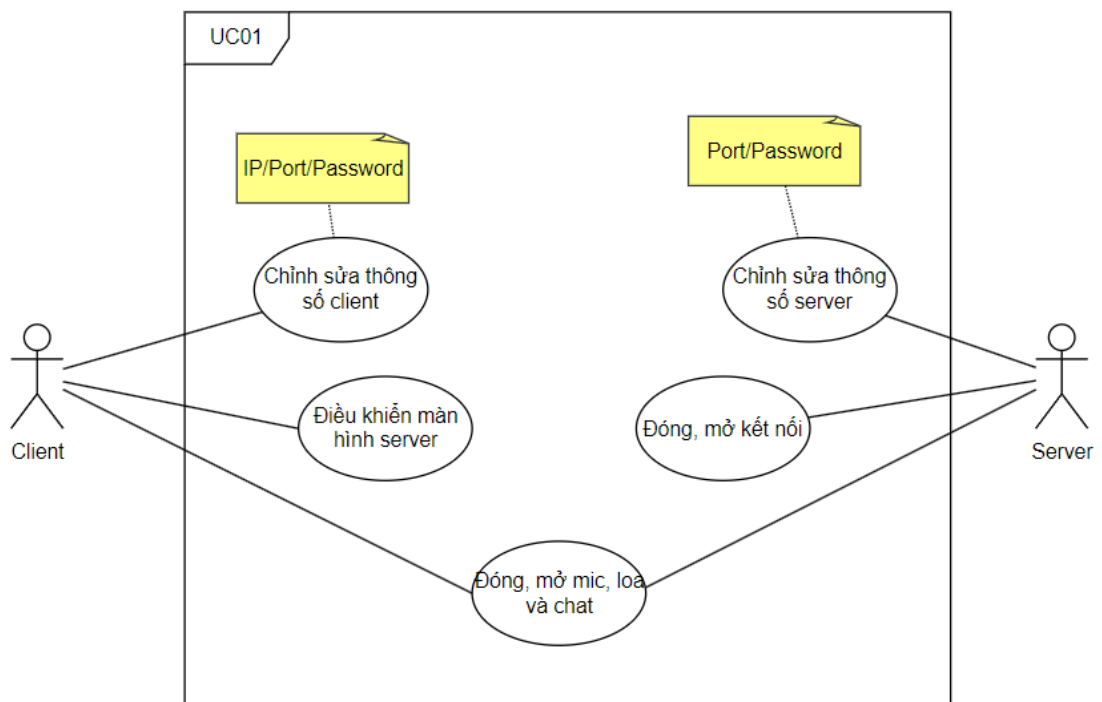
- Giao diện đơn giản, thân thiện với người dùng, dễ dàng thao tác sử dụng.
- Thời gian chạy nhanh, ổn định.
- Có khả năng mở rộng.

2.3. Mô tả chung về hệ thống

Hệ thống gồm 2 chương trình Client và Server. Máy tính điều khiển sẽ cài đặt chương trình Client và máy tính được điều khiển sẽ cài đặt chương trình Server.

Chương trình Server sẽ gửi tín hiệu màn hình của máy được điều khiển liên tục trong quá trình kết nối đến Client và nhận các tín hiệu điều khiển chuột, bàn phím từ Client.

Ở chiều ngược lại, chương trình Client nhận hình ảnh màn hình được đồng bộ theo thời gian từ server và gửi đi các tín hiệu điều khiển để điều khiển máy tính được điều khiển (server).



Hình 2.1: Sơ đồ use-case của hệ thống

2.4. Tổ chức mã nguồn

2.4.1. Client :

- Home Form: Chứa giao diện chương trình client lúc khởi động ban đầu, thực hiện chức năng kết nối và xác thực
- Main Form: Chứa giao diện khi thực hiện chức năng điều khiển máy tính, hiển thị màn hình của máy tính được điều khiển và các chức năng tắt bật âm thanh, mở chat form
- ChatForm: Thực hiện chức năng chat giữa client và server
- Class MyClient: Chức mã nguồn chính của chương trình client, cung cấp các hàm để chương trình thực hiện chức năng điều khiển máy tính
- Class HookKeyBoard: Chứa mã thực hiện chức năng chặn bắt sự kiện nhấn phím.

2.4.2. Server :

- MainForm: Chứa giao diện chương trình server lúc khởi động ban đầu, thực hiện chức năng cho phép client kết nối và xác thực, các chức năng tắt bật âm thanh, mở chat form, hiển thị các thông báo trong quá trình chạy chương trình
- ChatForm: Thực hiện chức năng chat giữa client và server
- Class MyServer: Chứa mã nguồn để tạo kết nối TCP, cho phép client kết nối và đóng kết nối.
- Class Client Handler: Chứa mã nguồn xử lý các yêu cầu đến từ client, cung cấp các hàm chính để chương trình server chia sẻ dữ liệu và nhận các tín hiệu điều khiển từ client

2.4.3 Thư viện tự tạo:

- Class DataHelper: Chứa các phương thức chuyển đổi kiểu dữ liệu qua lại với nhau
- Class InputEventCus: Chứa những đối tượng , phương thức và các khai báo chung cấp thấp để giả lập thao tác chuột và bàn phím trong hệ điều hành window.
- Class KeyboardCus: Chứa các phương thức xử lý sự kiện bàn phím (KeyUp, KeyDown).
- Class MouseCus: Chứa các phương thức xử lý sự kiện chuột.
- Class ScreenCus: Chứa các phương thức dùng để chụp màn hình và lấy kích thước màn hình.
- Class VoiceIn : Chứa các phương thức nhận tín hiệu âm thanh từ microphone, chuyển đổi tín hiệu âm thanh thành tín hiệu số, gửi tín hiệu đến host cụ thể
- Class VoiceOut: Chứa các phương thức nhận tín hiệu số từ host, chuyển đổi tín hiệu số thành tín hiệu âm thanh và phát ra loa.

2.5. Phân tích vận hành hệ thống

2.5.1 Truyền tín hiệu hình ảnh

Sau khi quá trình thiết lập kết nối hoàn tất, server sẽ truyền tín hiệu màn hình máy được điều khiển đến máy tính nắm quyền điều khiển bằng cách liên tục chụp ảnh màn hình của máy được điều khiển sau đó chuyển ảnh màn hình đó sang byte và cuối cùng truyền mảng byte đó đến với client (máy nắm quyền điều khiển). Việc chụp ảnh màn hình và gửi hình ảnh liên tục sẽ tạo ra dòng stream liên tục tạo thành chức năng chia sẻ màn hình của server cho client.

```
public static class ScreenCus{
    [StructLayout(LayoutKind.Sequential)]
    struct CURSORINFO{
        public Int32 cbSize;
        public Int32 flags;
        public IntPtr hCursor;
        public POINTAPI ptScreenPos; }
    [StructLayout(LayoutKind.Sequential)]
    struct POINTAPI {
        public int x;
        public int y; }
    [DllImport("user32.dll")]
    static extern bool GetCursorInfo(out CURSORINFO pci);
    [DllImport("user32.dll")]
    static extern bool DrawIcon(IntPtr hDC, int X, int Y, IntPtr hIcon);
    const Int32 CURSOR_SHOWING = 0x00000001;
    public static Bitmap CaptureScreen(bool CaptureMouse){
        Bitmap result = new Bitmap(Screen.PrimaryScreen.Bounds.Width,
        Screen.PrimaryScreen.Bounds.Height, PixelFormat.Format16bppRgb555);
        try{ using (Graphics g = Graphics.FromImage(result))
            {g.CopyFromScreen(0, 0, 0, 0, Screen.PrimaryScreen.Bounds.Size,
            CopyPixelOperation.SourceCopy);
                if (CaptureMouse){ CURSORINFO pci;
                    pci.cbSize =
                    System.Runtime.InteropServices.Marshal.SizeOf(typeof(CURSORINFO));
                    if (GetCursorInfo(out pci)){
                        if (pci.flags == CURSOR_SHOWING){
                            DrawIcon(g.GetHdc(), pci.ptScreenPos.x, pci.ptScreenPos.y,
                            pci.hCursor);
                            g.ReleaseHdc();}
                        }}}
            }
        catch{result = null; }
        return result;
    }
    public static Point GetScreenSize(){
        return new Point(Screen.PrimaryScreen.Bounds.Width,
        Screen.PrimaryScreen.Bounds.Height); }}

```

Hình 2.2: Đoạn mã thiết lập chức năng chụp ảnh màn hình

2.5.2 Truyền tín hiệu điều khiển chuột

Tín hiệu chuột được truyền từ client sang server để khi client thực hiện các thao tác chuột trên màn hình điều khiển thì máy tính được điều khiển sẽ nhận được các thao tác tương ứng và tiến hành thực hiện các thao tác đó. Tín hiệu chuột được gửi đi bao gồm tọa độ của con trỏ chuột trên màn hình và thao tác chuột được thực hiện.

Ở đây, chương trình thực hiện tính toán tỉ lệ của màn hình client so với màn hình server, sau đó, khi độ dời con trỏ chuột ở màn hình điều khiển (client) thay đổi thì độ dời con trỏ chuột ở màn hình của máy tính được điều khiển (server) sẽ thay đổi tương ứng với tỉ lệ đó. Các thông số được gửi đi là tọa độ X, Y của con trỏ chuột trên màn hình điều khiển (client), server sẽ nhận các tọa độ đó, quy đổi ra tọa độ tương ứng theo tỉ lệ màn hình sau đó con trỏ chuột ở màn hình được điều khiển (server) sẽ di chuyển đến vị trí tương ứng. Các thao tác với chuột khác cũng sẽ được truyền với nguyên tắc tương tự. Sẽ có một Enum lưu mã của các thao tác chuột, sau đó, khi client thực hiện thao tác, chương trình sẽ gửi mã của thao tác tương ứng đến server và chương trình server sẽ thực hiện thao tác đó trên máy được điều khiển.

```
public enum ClientMessage
{
    MOUSE_MOVE = 0,
    MOUSE_LEFT_DOWN,
    MOUSE_LEFT_UP,
    MOUSE_RIGHT_DOWN,
    MOUSE_RIGHT_UP,
    MOUSE_SCROLL,
    KEY_UP,
    KEY_DOWN,
    MESSAGE,
}
```

Hình 2.3: Bảng mã các thao tác chuột

```

public static void MouseMove(int x, int y){
    Input[] inputs = new Input[1];
    inputs[0].type = (int)InputType.Mouse;
    inputs[0].u.mi.dwFlags = (uint)MouseEventF.Move | (uint)MouseEventF.Absolute;
    inputs[0].u.mi.dx = x * 65535 / Screen.PrimaryScreen.Bounds.Width;
    inputs[0].u.mi.dy = y * 65535 / Screen.PrimaryScreen.Bounds.Height;
    SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
}

public static void MouseDown(int x, int y, MouseEventF mouseEventF = MouseEventF.LeftDown){
    Input[] inputs = new Input[1];
    // Nhấn chuột trái xuống
    inputs[0].type = (int)InputType.Mouse;
    inputs[0].u.mi.dwFlags = (uint)mouseEventF;
    SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
}

public static void MouseUp(int x, int y, MouseEventF mouseEventF = MouseEventF.LeftUp){
    Input[] inputs = new Input[1];
    // Nhấn chuột trái xuống
    inputs[0].type = (int)InputType.Mouse;
    inputs[0].u.mi.dwFlags = (uint)mouseEventF;
    SendInput((uint)inputs.Length, inputs, Marshal.SizeOf(typeof(Input)));
}

public static void MouseScroll(int scrollValue){
    Input[] i = new Input[1];
    i[0].type = INPUT_MOUSE;
    i[0].u.mi.mouseData = (uint)scrollValue;
    i[0].u.mi.dwFlags = MOUSEEVENTF_WHEEL;
    SendInput((uint)i.Length, i, Marshal.SizeOf(i[0].GetType()));
}

[DllImport("user32.dll")]
public static extern bool GetCursorPos(out POINT lpPoint);
[StructLayout(LayoutKind.Sequential)]
public struct POINT
{
    public int X;
    public int Y;
}

[DllImport("User32.dll")]
public static extern bool SetCursorPos(int x, int y);

```

Hình 2.4: Các hàm xử lý các thao tác chuột

2.5.3 Truyền tín hiệu điều khiển bàn phím

Tính hiệu bàn phím được truyền từ client tới server theo mã của phím được ấn. Khi một phím được client ấn, chương trình client sẽ gửi một tính hiệu KeyDown với mã của phím tương ứng đến server, chương trình server tiến hành thao tác KeyDown tương ứng trên máy được điều khiển, sau đó, khi phím được nhả ra, client sẽ gửi tính hiệu KeyUp với mã của phím tương ứng đến server, server cũng sẽ tiến hành thao tác KeyUp tương ứng trên máy được điều khiển và hoàn

thành thao tác ấn phím. Một số phím đặc biệt mà khi bấm sẽ ảnh hưởng đến máy client như phím WinDow hay tổ hợp phím ALT+ TAP, do đó, khi khởi động chương trình Điều Khiển Máy Tính Từ Xa, chương trình sẽ tạo ra một Hook nhằm chặn bắt tất cả các thao tác phím mà người dùng thực hiện khi đang focus vào màn hình điều khiển, nói cách khác, khi người điều khiển đang focus vào màn hình điều khiển của ứng dụng, các phím mà người điều khiển bấm sẽ được chương trình chặn bắt và gửi đến máy được điều khiển(server) thực hiện, các phím đó sẽ không được thực hiện trên máy điều khiển (client). Khi người dùng bỏ focus vào màn hình điều khiển, các thao tác bấm phím trên máy tính điều khiển sẽ được thực hiện như bình thường.

```
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
{
    if (nCode >= 0)
    {
        KBDLLHOOKSTRUCT kbd = (KBDLLHOOKSTRUCT)Marshal.PtrToStructure(lParam,
        typeof(KBDLLHOOKSTRUCT));

        if (wParam == (IntPtr)WM_KEYDOWN)
        {
            Console.WriteLine(kbd.key.ToString() + " Down");
            SendKeyDown(kbd.key);
        }
        else if (wParam == (IntPtr)WM_KEYUP){
            Console.WriteLine(kbd.key.ToString() + " Up");
            SendKeyUp(kbd.key);
        }
        else if (!IsAltKeyUp(kbd.flags) && kbd.key == Keys.Tab){
            Console.WriteLine("alt Down + tab");
            SendKeyDown(Keys.LMenu);
            SendKeyDown(Keys.Tab);
        }
        else if (!IsAltKeyUp(kbd.flags)){
            Console.WriteLine("alt Down");
            SendKeyDown(Keys.LMenu);
        }
        return (IntPtr)1; // Chặn sự kiện
    }
    return CallNextHookEx(_hookID, nCode, (int)wParam, lParam);
}
```

Hình 2.5: Đoạn mã để Hook sự kiện bàn phím

2.5.4 Truyền tín hiệu âm thanh

Khi người dùng nói, tín hiệu âm thanh được máy tính của người nói nhận và chuyển tín hiệu đó sang mảng byte sử dụng hàm `RecorderOnDataAvailable` của thư viện `NAudio.Wave`.

```
public class VoiceIn
{
    private WaveInEvent waveIn;
    private UdpClient clientIn;
    private string hostname;
    private int port;
    public VoiceIn(string hostname, int port = 5920){
        clientIn = new UdpClient();
        waveIn = new WaveInEvent();
        waveIn.DataAvailable += RecorderOnDataAvailable;
        waveIn.WaveFormat = new WaveFormat(44100, 1);
        this.hostname = hostname;
        this.port = port;
    }
    public void StartRecording()
    {
        waveIn.StartRecording();
    }
    public void StopRecording()
    {
        waveIn?.StopRecording();
    }
    public WaveFormat GetWaveFormat()
    {
        return waveIn.WaveFormat;
    }
    public void Close(){
        if (clientIn != null){
            clientIn.Close();
        }
    }
    private void RecorderOnDataAvailable(object sender, WaveInEventArgs e)
    {
        try
        {
            clientIn.Send(e.Buffer, e.BytesRecorded, new IPEndPoint(IPAddress.Parse(hostname),
port));
        }
        catch (Exception) {}
    }
}
```

Hình 2.6: Đoạn mã nhận tín hiệu âm thanh vào

Mảng byte tín hiệu tiếng nói sau đó được truyền sang máy người nhận, sau đó, chương trình tiếp tục biến đổi mảng byte đó thành tín hiệu âm thanh và phát ra loa.

```
public class VoiceOut
{
    private UdpClient clientOut;
    private WaveOutEvent waveOut;
    private BufferedWaveProvider bufferedWaveProvider;
    public VoiceOut(WaveFormat waveFormat, int port = 5920){
        clientOut = new UdpClient(port);
        bufferedWaveProvider = new BufferedWaveProvider(waveFormat);
        waveOut = new WaveOutEvent();
        waveOut.Init(bufferedWaveProvider);
    }
    public void Close(){
        if (clientOut != null){
            clientOut.Close();}
    }
    public void Play(){
        waveOut.Play();
    }
    public void ReceiveData(){
        try{
            IPEndPoint endPoint = null;
            var data = clientOut.Receive(ref endPoint);
            bufferedWaveProvider.AddSamples(data, 0, data.Length);
        }
        catch (Exception) {}
    }
    public void Stop(){
        waveOut?.Stop();
    }
}
```

Hình 2.7: Đoạn mã phát tín hiệu âm thanh

CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Môi trường cài đặt:

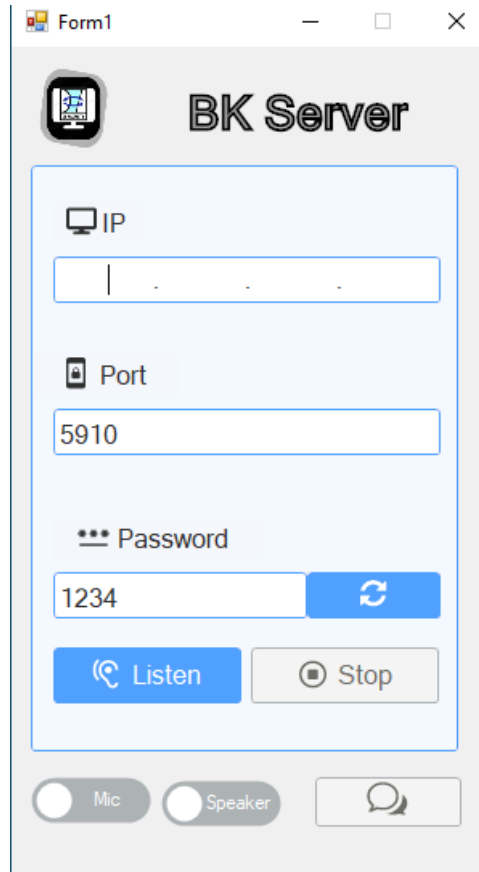
- Ngôn ngữ lập trình: C#.
- Hệ điều hành: Windows.
- Môi trường phát triển: Microsoft Visual Studio
- Framework: Winform.

Mã nguồn chương trình

1. Chương trình client : <https://github.com/titentam/ClientPBL4>
2. Chương trình server: <https://github.com/titentam/ServerPBL4>
3. Thư viện tự xây dựng: <https://github.com/titentam/ATiLibraryControlCSharp>

3.2. Giao diện hệ thống

3.2.1. Giao diện chương trình server

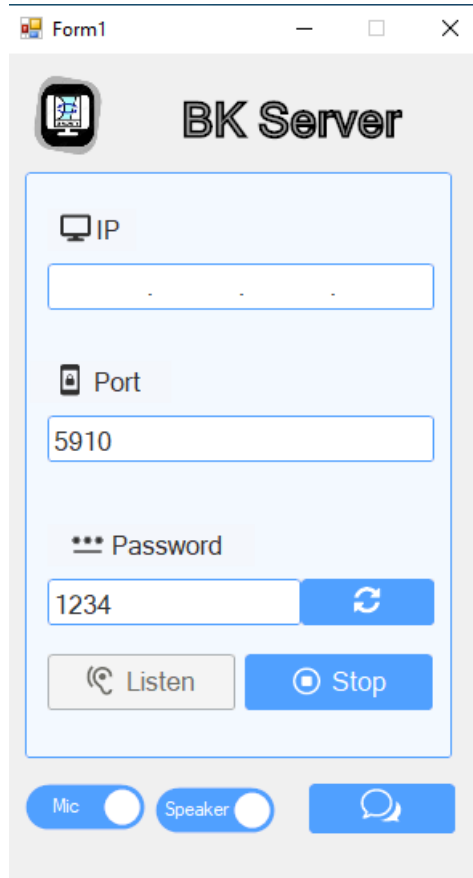


Hình 3.1: Giao diện chương trình server

Khi khởi động chương trình server, cửa sổ chương trình hiện lên giao diện như trên.

Trong đó, địa chỉ ip là địa chỉ ip hiện tại của máy, Port là cổng đang lắng nghe kết nối, ở đây chương trình chạy cổng mặc định là 5910, người dùng có thể tùy chỉnh cổng phù hợp với máy của mình. Password do chương trình tạo ngẫu nhiên, người dùng có thể tạo password mới bằng cách ấn nút “Regenarate” bên cạnh ô password.

Người dùng ấn vào nút listen để chương trình lắng nghe các kết nối từ các client gửi kết nối. Sau khi đã hoàn thành công việc, nhấn Stop để ngừng nhận kết nối.



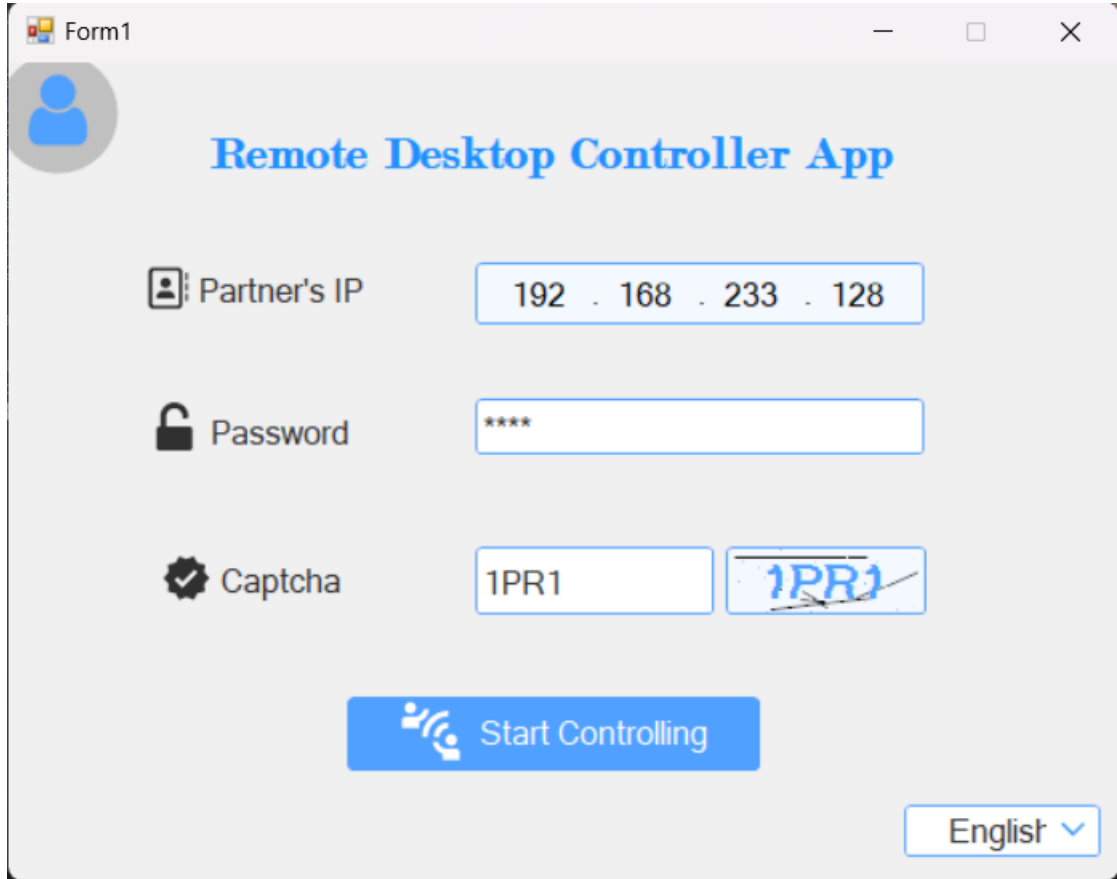
The image shows a Windows application window titled "Form1". Inside the window is a panel titled "BK Server" with a server icon. The panel contains several input fields and buttons:

- An "IP" input field with a computer monitor icon.
- A "Port" input field with a padlock icon, containing the value "5910".
- A "Password" input field with three asterisks, containing the value "1234", and a blue button with a refresh icon.
- A "Listen" button with a microphone icon and a "Stop" button with a stop icon.
- At the bottom, there are three buttons: "Mic" with a microphone icon, "Speaker" with a speaker icon, and "Chat" with a speech bubble icon.

Hình 3.2: Server đang lắng nghe kết nối và bật kết nối âm thanh

Người dùng có thể tùy chọn phương thức giao tiếp bằng âm thanh hoặc bằng cách nhắn tin thông qua việc ấn các nút “Mic”, “Speaker” hoặc “Chat”.

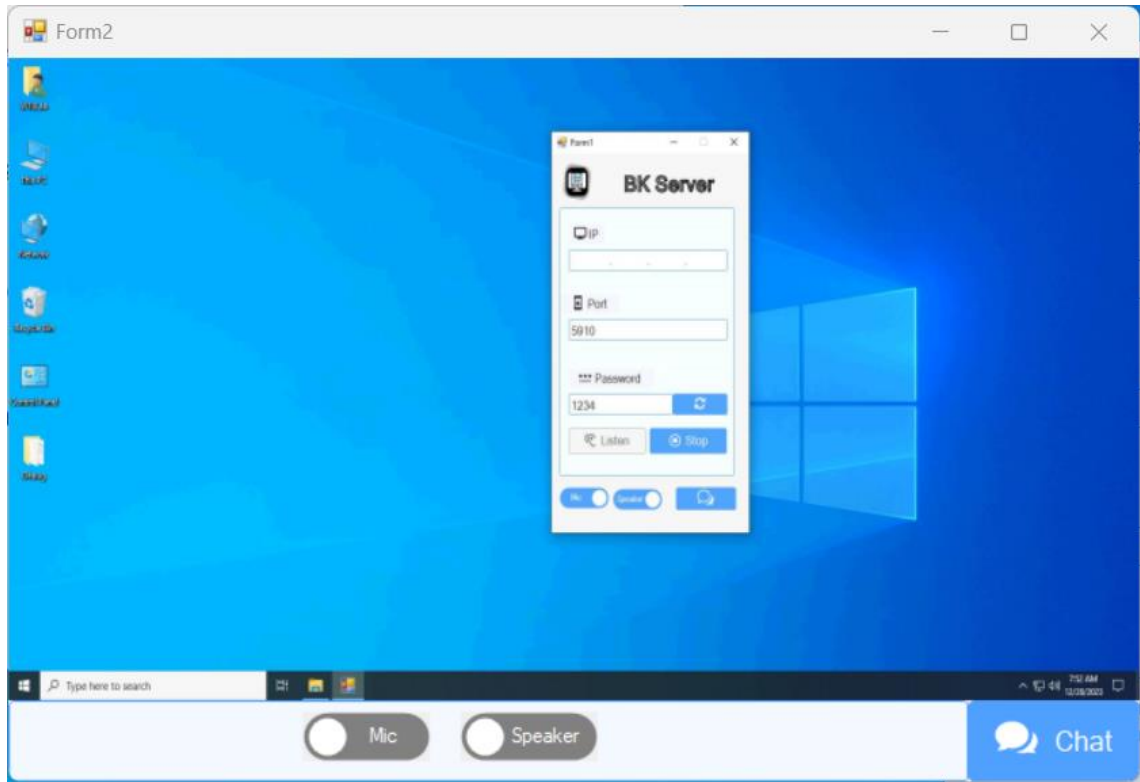
3.2.2. Giao diện chương trình client

The image shows a Windows application window titled "Form1". Inside the window, there is a user icon in the top left corner. The main title "Remote Desktop Controller App" is displayed in blue text. Below the title, there are three input fields: "Partner's IP" with the value "192 . 168 . 233 . 128", "Password" with masked characters "****", and "Captcha" with the value "1PR1". To the right of the "Captcha" input is a small image of the captcha "1PR1". Below these fields is a large blue button labeled "Start Controlling" with a wireless signal icon. In the bottom right corner, there is a language dropdown menu currently set to "English".

Hình 3.3: Giao diện chương trình client

Khi khởi động chương trình client, màn hình chính của chương trình client sẽ hiển thị. Người điều khiển cần nhập đúng địa chỉ ip của máy được điều khiển (server), mật khẩu (do server cung cấp) và xác thực mã captcha. Sau đó, người dùng có thể bắt đầu điều khiển máy server bằng cách nhấn “Start Controlling” (với điều kiện là server đã sẵn sàng lắng nghe kết nối).

Ngoài ra, người dùng có thể thay đổi ngôn ngữ hiển thị từ Tiếng Anh sang Tiếng Việt nếu muốn bằng cách chọn ngôn ngữ ở góc dưới bên phải cửa sổ chương trình.

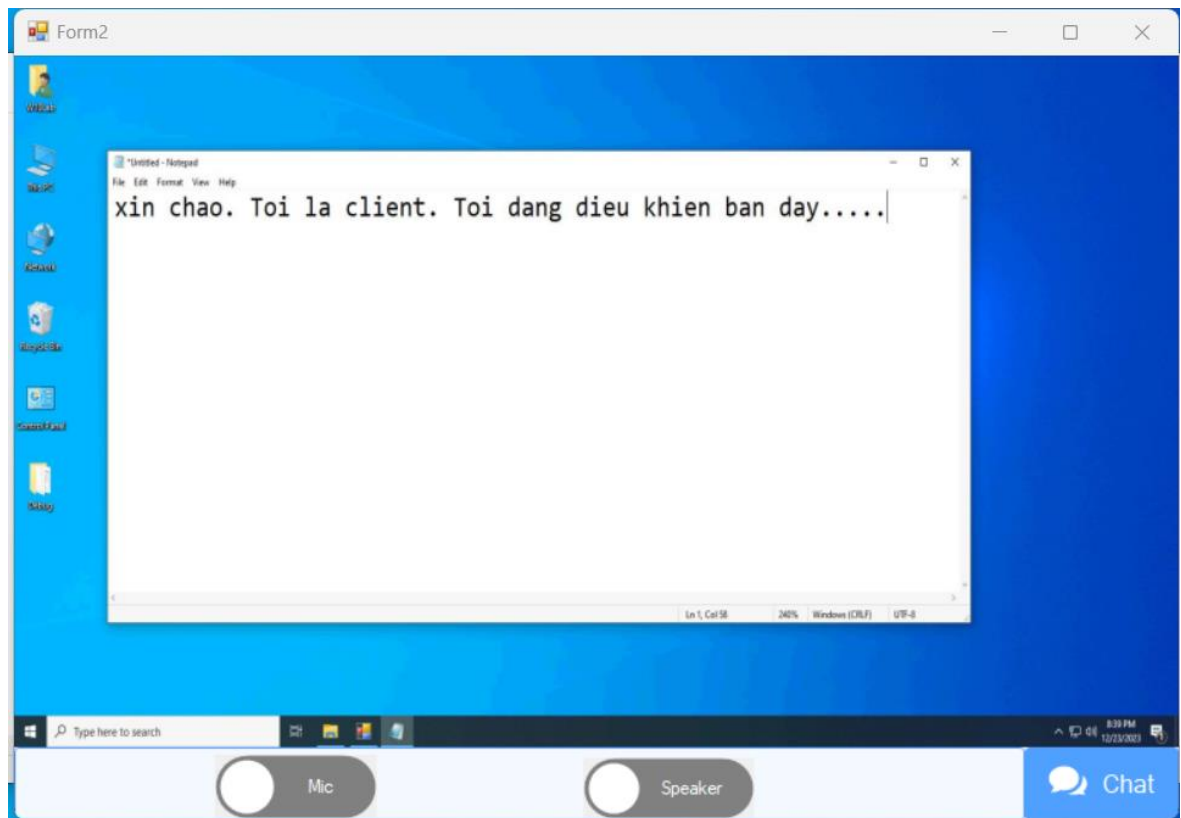


Hình 3.4: Giao diện màn hình điều khiển

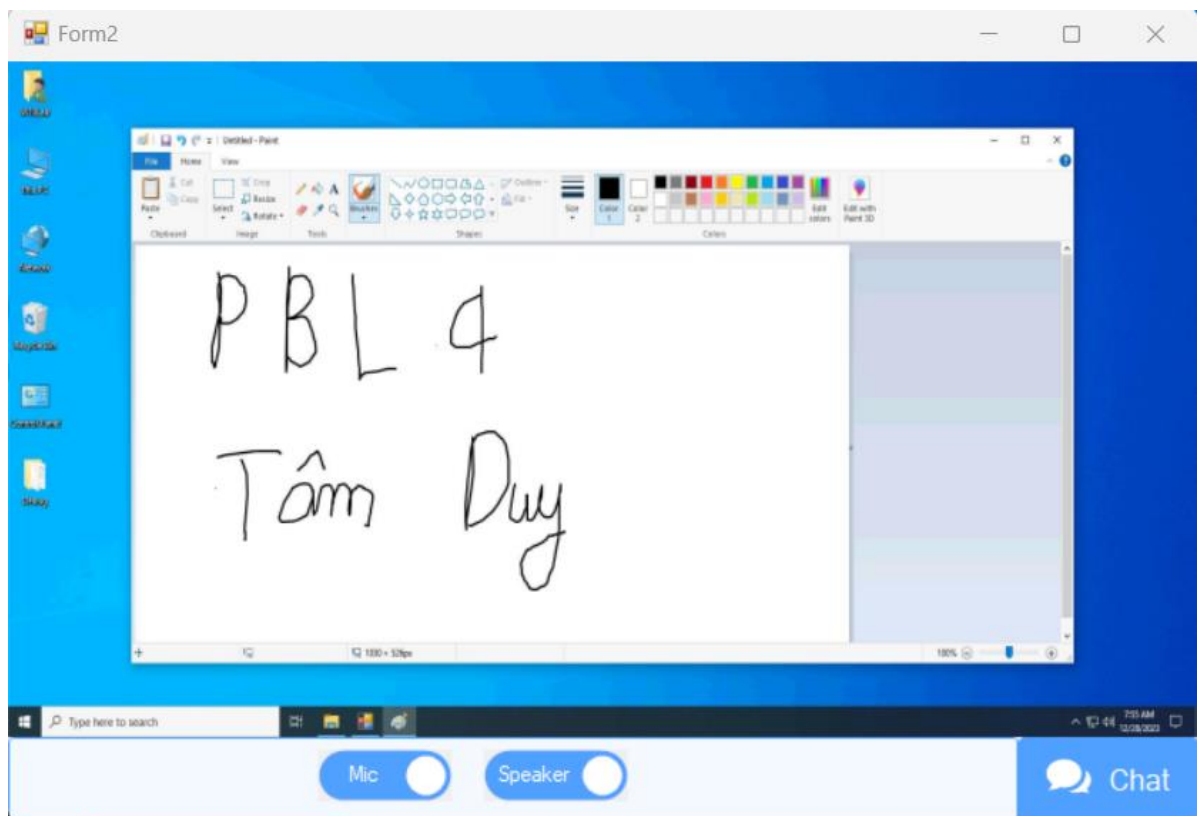
Sau khi người dùng nhấn “Start Controlling” ở giao diện chính, màn hình điều khiển của chương trình sẽ xuất hiện. Người dùng có thể nhìn thấy giao diện màn hình theo thời gian thực của máy được điều khiển(server) và có thể thao tác điều khiển thông qua màn hình này.

Người điều khiển có thể tùy chọn phương thức giao tiếp với đối phương bằng âm thanh hoặc bằng tin nhắn thông qua các thao tác bật tắt “Mic”, “Speaker” hoặc “Chat”.

Người điều khiển có thể thực hiện các thao tác với bàn phím, chuột ở máy được điều khiển tương tự như thao tác trên máy của mình.

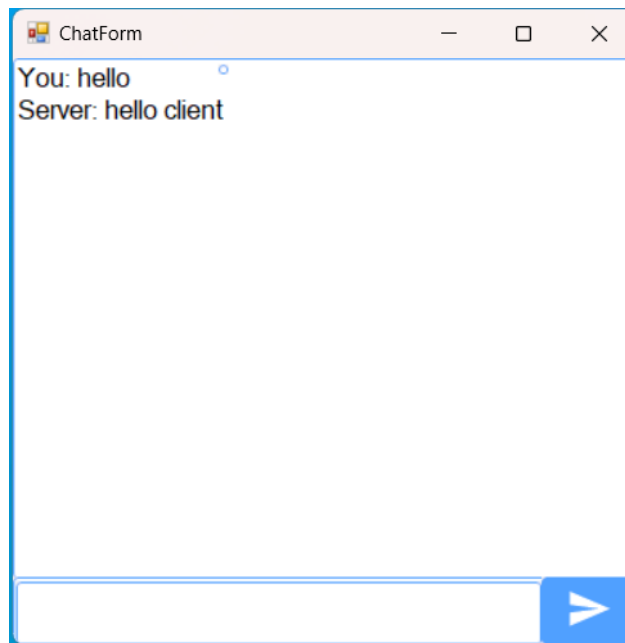


Hình 3.5: Thực hiện các thao tác điều khiển với bàn phím

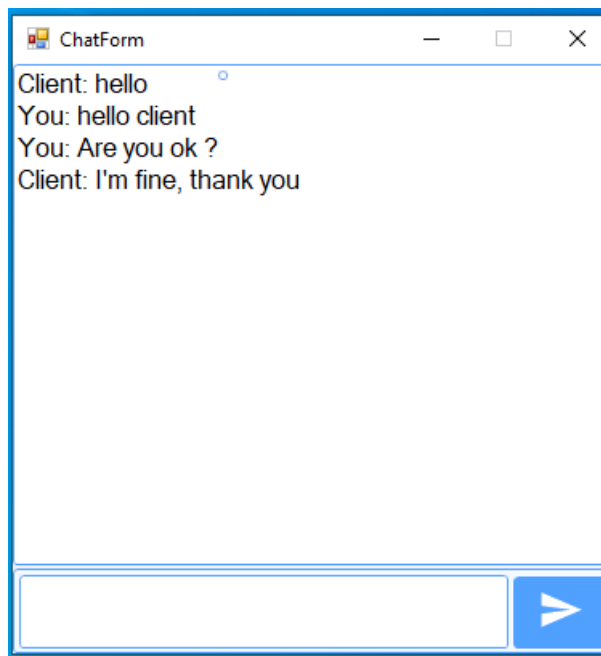


Hình 3.6: Thực hiện thao tác điều khiển với chuột và mở kết nối âm thanh

Ngoài các chức năng điều khiển chính ra, người dùng có thể sử dụng chức năng “Chat” để giao tiếp giữa client và server trong trường hợp gặp sự cố về thiết bị âm thanh.



Hình 3.7: Giao tiếp bằng chức năng "Chat"(giao diện Client)



Hình 3.8:Giao tiếp bằng chức năng "Chat"(giao diện Server)

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận:

Ứng dụng điều khiển máy tính từ xa đã được triển khai đúng như mô tả kỹ thuật. Dữ liệu được trao đổi giữa server và client ổn định, không xuất hiện tình trạng giật lag, sai tín hiệu

Đã hoàn thành các yêu cầu về đề tài:

- Giao diện đơn giản, thân thiện với người dùng, dễ dàng thao tác sử dụng.
- Sử dụng trên nền tảng ứng dụng máy tính window.
- Cho phép điều khiển được máy tính khác từ xa, dữ liệu được truyền nhanh chóng, chính xác, độ phân giải màn hình ổn định.

2. Hướng phát triển:

- Nghiên cứu phát triển để dùng được cho mạng internet thay vì mạng nội bộ.
- Cải thiện giao diện, tăng trực quan cho người dùng.
- Thêm tùy chọn đăng nhập cho người dùng để thực hiện các chức năng tăng tính bảo mật cho hệ thống.
- Nghiên cứu cơ chế nén dữ liệu để tăng độ phân giải màn hình mà vẫn đảm bảo độ mượt.
- Thêm cơ chế kiểm tra và mã hóa dữ liệu để tăng tính bảo mật cho dữ liệu hệ thống.

TÀI LIỆU THAM KHẢO

1. Slide môn “Lập trình mạng” - Thầy Mai Văn Hà - Bách Khoa Đà Nẵng.
2. Slide môn “Nguyên lí hệ điều hành” - Cô Trần Hồ Thủy Tiên- Trường ĐH Bách Khoa Đà Nẵng.
3. T.RichardSon, J.Levine, “The Remote Framebuffer Protocol”,
<https://datatracker.ietf.org/doc/html/rfc6143>, [7/10/2023].
4. Bojidar Qnkov, “How to Send Inputs using C#”,
https://www.codeproject.com/Articles/5264831/How-to-Send-Inputs-using-Csharp?fbclid=IwAR3pabffe6DFw79RAx6IPXJieBedtOIg_vm18O4fOMr9sumGyddG-tD5-U, [18/10/2023].
5. Mark Heath, “NAudio .Net Library”,
<https://github.com/naudio/NAudio> , [20/11/2023].

PHỤ LỤC

Phiên bản .Net: .Net Framework 4.8.

Mã nguồn minh họa:

1. Class chính xử lý ở chương trình client

```
public class MyClient
{
    private TcpClient client;
    private NetworkStream stream;
    private BinaryReader reader;
    private BinaryWriter writer;
    private int port;
    private VoiceIn voiceIn;
    private VoiceOut voiceOut;
    private string ipServer;
    private HookKeyboard hook;
    private bool isListening; // voice chat from server

    public bool isConnected { get => client.Connected;}

    public MyClient(string ipServer, int port)
    {
        this.ipServer = ipServer;
        this.port = port;
    }

    public void Close()
    {
        if(client != null)
        {
            client.Close();
        }
        if(voiceIn != null)
        {
            voiceIn.Close();
        }
        if( voiceOut != null )
        {
            voiceOut.Close();
        }
    }

    public void Connect()
    {
        client = new TcpClient();
        client.Connect(ipServer, port);
        stream = client.GetStream();
        reader = new BinaryReader(stream);
        writer = new BinaryWriter(stream);
    }
}
```

```

InitVoice();

hook = HookKeyBoard.getInstance(stream, writer);
}

private void InitVoice()
{

    voiceIn = new VoiceIn(ipServer, 6969);
    voiceOut = new VoiceOut(voiceIn.GetWaveFormat(), 6969);

}

public void ReceiveVoice()
{
    voiceOut.Play();
    islistening = true;
    Thread t = new Thread(() =>
    {
        while (islistening)
        {
            voiceOut.ReceiveData();
        }
    });
    t.Start();

}

public void StopReceiveVoice()
{
    islistening = false;
    voiceOut.Stop();
}

public void VoiceRecorder()
{
    Thread t = new Thread(() =>
    {
        voiceIn.StartRecording();
    });
    t.Start();
}

public void VoiceStop()
{
    voiceIn.StopRecording();
}

public bool SendPass(string pass)
{
    if(isConnected)
    {
        try
        {
            writer.Write(pass);
            return reader.ReadBoolean();
        }
    }
}

```



```

    }
    catch (Exception ex)
    {
        return false;
    }

    }
    return false;
}

public void ReceiveScreenDesktop(ref PictureBox screen, ref bool isClosed)
{
    while (client.Connected)
    {
        try
        {
            int length = reader.ReadInt32();

            byte[] bitmapByte = reader.ReadBytes(length);

            var bitmap = DataHelper.ByteArrayToBitmap(bitmapByte);
            screen.Image = bitmap;
        }
        catch (IOException ex)
        {
            MessageBox.Show("Disconnected!");
            isClosed = true;
            break;
        }
    }
}

public void ReceiveMessage(Action<string> AppendMessage)
{
    while (client.Connected)
    {
        string message = reader.ReadString();
        AppendMessage("Server: " + message);
    }
}

public void SendMessage(string message)
{
    if (isConnected)
    {
        writer.Write((byte)ControlCustom.ClientMessage.MESSAGE);
        writer.Write(message);
        stream.Flush();
    }
}
}

```

```

public void SendMouseMove(double scaleX, double scaleY)
{
    if (isConnected){
        try{
            writer.Write((byte)ControlCustom.ClientMessage.MOUSE_MOVE);
            writer.Write(scaleX);
            writer.Write(scaleY);
            stream.Flush();
        }
        catch(Exception ex)
        {
            return;
        }
    }

}

public void SendMouseScroll(int scrollValue)
{
    if (isConnected)
    {
        try
        {
            writer.Write((byte)ControlCustom.ClientMessage.MOUSE_SCROLL);
            writer.Write(scrollValue);
            stream.Flush();
        }
        catch (Exception ex)
        {
            return;
        }
    }

}

public void SendMouseDown(double scaleX, double scaleY, MouseButton btn = MouseButton.Left)
{
    if (isConnected)
    {
        try
        {
            byte type;
            if (btn == MouseButton.Left)
            {
                type = (byte)ControlCustom.ClientMessage.MOUSE_LEFT_DOWN;
            }
            else
            {
                type = (byte)ControlCustom.ClientMessage.MOUSE_RIGHT_DOWN;
            }

            writer.Write(type);
        }
        catch (Exception ex)
        {
            return;
        }
    }
}

```

PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH

```
        writer.Write(scaleX);
        writer.Write(scaleY);
        stream.Flush();
    }
    catch( Exception ex)
    {
        return;
    }
}

}

public void SendMouseUp(double scaleX, double scaleY, MouseButton btn = MouseButton.Left)
{
    if(isConnected)
    {
        try
        {
            byte type;
            if (btn == MouseButton.Left)
            {
                type = (byte)ControlCustom.ClientMessage.MOUSE_LEFT_UP;
            }
            else{
                type = (byte)ControlCustom.ClientMessage.MOUSE_RIGHT_UP;
            }
            writer.Write(type);
            writer.Write(scaleX);
            writer.Write(scaleY);
            stream.Flush();
        }
        catch (Exception)
        {
            return;
        }
    }
}

public void SetHook()
{
    hook.SetHook();
}

public void UnHook()
{
    if (HookKeyBoard._hookID != IntPtr.Zero)
    {
        HookKeyBoard.UnhookWindowsHookEx(HookKeyBoard._hookID);
    }
}

}
```

2.Hàm chụp màn hình và gửi ảnh màn hình ở chương trình server

```
public void SendDesktop(int fps = 60){
    while (isConnected){
```

```

        try{
            var bitmap = ScreenCus.CaptureScreen(false);

            byte[] bitmapBytes = DataHelper.BitmapToByteArray(bitmap);

            writer.Write(bitmapBytes.Length);

            writer.Write(bitmapBytes);

            stream.Flush();
            Thread.Sleep(1000 / fps);
        }
        catch (Exception){
            MessageBox.Show("Client had disconnected!");
            this.Stop();
            break;
        }
    }
}

```

3.Hàm xử lí các tính hiệu điều khiển nhận từ client của chương trình server

```

public void ReceiveClientMessage(){
    while (isConnected){
        ClientMessage clientMessage;
        try{
            clientMessage = (ClientMessage)reader.ReadByte();
        }
        catch (Exception){
            this.Stop();
            break;
        }
        switch (clientMessage){
            case ClientMessage.MOUSE_MOVE:
            {
                int x = 0, y = 0;
                ReceivePosition(ref x, ref y);
                MouseCus.MouseMove(x, y);
                break;
            }
        }
    }
}

```

```

    }

    case ClientMessage.MOUSE_LEFT_DOWN:
    {
        int x = 0, y = 0;
        ReceivePosition(ref x, ref y);
        MouseCus.MouseDown(x, y);
        break;
    }

    case ClientMessage.MOUSE_LEFT_UP:
    {
        int x = 0, y = 0;
        ReceivePosition(ref x, ref y);
        MouseCus.MouseUp(x, y);
        break;
    }

    case ClientMessage.MOUSE_RIGHT_DOWN:
    {
        int x = 0, y = 0;
        ReceivePosition(ref x, ref y);
        MouseCus.MouseDown(x, y,
InputEventCus.MouseEventF.RightDown);
        break;
    }

    case ClientMessage.MOUSE_RIGHT_UP:
    {
        int x = 0, y = 0;
        ReceivePosition(ref x, ref y);
        MouseCus.MouseUp(x, y,
InputEventCus.MouseEventF.RightUp);
        break;
    }

    case ClientMessage.MOUSE_SCROLL:
    {
        int scrollValue = reader.ReadInt32();
        MouseCus.MouseScroll(scrollValue);
        break;
    }

    case ClientMessage.KEY_UP:
    {
        int keyValue = reader.ReadInt32();

```

```
        Keys key = (Keys)keyValue;
        KeyBoardCus.KeyUp(key);
        break;
    }
    case ClientMessage.KEY_DOWN:
    {
        int keyValue = reader.ReadInt32();
        Keys key = (Keys)keyValue;
        KeyBoardCus.KeyDown(key);
        break;
    }
    case ClientMessage.MESSAGE:
    {
        string message = reader.ReadString();

        break;
    }
    default:
        break;
}}
}
```