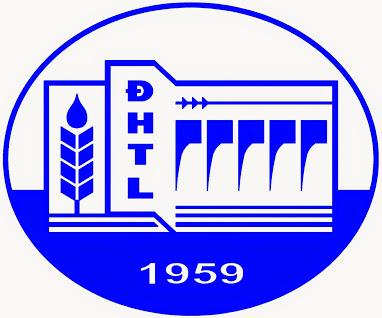
**TRƯỜNG ĐẠI HỌC THỦY LỢI  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỆ ĐIỀU HÀNH**

**ĐỀ TÀI: SỬ DỤNG HÀM API XÂY DỰNG ỨNG DỤNG HẸN GIỜ TẮT MÁY TÍNH VÀ TỰ ĐỘNG TẮT MÁY KHI HẾT PIN**

**Giảng viên hướng dẫn: TS. Phạm Thanh Bình**

**Nhóm sinh viên thực hiện:**

1. **Hoàng Thị Thu Uyên – 61TH4**
2. **Hoàng Minh Quang – 61TH4**
3. **Vũ Thế Song – 61TH4**

**Mục lục**

Contents

[Mở đầu 2](#_Toc101986027)

[1. Giới thiệu 2](#_Toc101986028)

[2. Nội dung báo cáo 3](#_Toc101986029)

[Chương 1: Tổng quan về hàm API và tiến trình trên hệ điều hành windows 3](#_Toc101986030)

[1.1 Hàm API trên hệ điều hành windows 3](#_Toc101986031)

[1.1.1 Hàm API là gì? 3](#_Toc101986032)

[1.1.2 Một số hàm API trên hệ điều hành windows: 3](#_Toc101986033)

[1.1.2.1 Các API được tổ chức trong 4DLL của windows 3](#_Toc101986034)

[1.1.2.2. Một số hàm API 4](#_Toc101986035)

[1.2 Tổng quan về tiến trình 5](#_Toc101986036)

[1.2.1 Tiến trình là gì? 5](#_Toc101986037)

[1.2.2 Chương trình ví dụ về tiến trình 5](#_Toc101986038)

[Chương 2: Chương trình ứng dụng hẹn giờ tự tắt máy tính và tự động tắt máy khi hết pin 6](#_Toc101986039)

[2.1 Giới thiệu bài toán và chương trình ứng dụng 6](#_Toc101986040)

[2.2 Mã nguồn chương trình ứng dụng và thực nghiệm 7](#_Toc101986041)

[2.2.1 Mã nguồn chương trình 7](#_Toc101986042)

[2.2.2 Hình ảnh thực nghiệm chương trình 13](#_Toc101986043)

[Chương 3: Kết luận và hướng phát triển 15](#_Toc101986044)

[3.1 Kết luận. 15](#_Toc101986045)

[3.2 Tài liệu tham khảo 15](#_Toc101986046)

# Mở đầu

## Giới thiệu

Hệ điều hành là một hệ chương trình hoạt động giữa người dùng và phần cứng của máytính. Từ đó, cung cấp một môi trường để người sử dụng có thể thi hành các chương trìnhlàm cho máy tính dễ sử dụng hơn, thuận tiện và hiệu quả.Để giúp người dùng tắt máy tính 1 cách nhanh chóng như trước khi ngủ mà k bị tỉnh giấc vì những thao tác để tắt máy tính nhóm chúng em đã chọn đề tài “Ứng dụng dùng hàm API để hẹn giờ tự động tắt máy tính”

Công cụ sử dụng

Ngôn ngữ lập trình: C#

Phần mềm code: Visual studio 2019

## Nội dung báo cáo

* Chương 1: Tổng quan về hàm API và tiến trình trên hệ điều hành window
* Chương 2: Chương trình ứng dụng hẹn giờ tắt máy tính và tự động tắt máy khi hết pin
* Chương 3: Kết luận và hướng phát triển

# Chương 1: Tổng quan về hàm API và tiến trình trên hệ điều hành windows

## 1.1 Hàm API trên hệ điều hành windows

### 1.1.1 Hàm API là gì?

API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng)phương thức kết nối với các thư viện và ứng dụng khác. Nó chính là một phần mềm giaotiếp được sử dụng bởi các ứng dụng khác nhau giữa chương trình và hệ điều hành.

### 1.1.2 Một số hàm API trên hệ điều hành windows:

#### 1.1.2.1 Các API được tổ chức trong 4DLL của windows

**1.1.2.1.1 KERNEL32:**

Là DLL chính, đảm nhiệm quản lý bộ nhớ, thực hiện chức năng đa nhiệm và những hàm ảnh hưởng trực tiếp đến hoạt động của windows

**1.1.2.1.2 USER32:**

Thư viện quản lý Windows. Thư viện này chứa các hàm xử lý menu, định giờ, truyền tin,tập tin và nhiều phần không được hiển thị kháccủa Windows.

**1.1.2.1.3 GDI32**

Giao diện thiết bị đồ hoạ (Graphics Device Interface). Thư viện này cung cấp các hàm vẽtrên màn hình, cũng như kiểm tra phần biểu mẫu nào cần vẽ lại.

**1.1.2.1.4 WINNM**

Cung cấp các hàm multimedia để xử lý âm thanh, nhạc, video thời gian thực, lấy mẫu, v.v…Nó là DLL 32 bit. (Thư viện 16bit tên là MMSYSTEM)Ta có thể tìm các tập tin này trong thư mục \Windows\system. Ngoài ra, còn có các DLLnhỏ hơn, cũng được dùng phổ biến để cung cấp các dịch vụ đặc biệt cho ứng dụng.Trên đây là các tên DLL 32 bit. Phiên bản VB4 là bản cuối cùng còn hỗ trợ 16 bit.

#### 1.1.2.2. Một số hàm API

* Declare Function AnyPopup Lib "user32" Alias "AnyPopup" () As LongCông dụng: Đưa ra chỉ số cửa sổ popup hiện đang tồn tại trên màn hình.Trị trả về: Integer ~ True (Khác zero) nếu có cửa sổ popup.
* Declare Function AdjustWindowRectEx Lib "user32" Alias "AdjustWindowRectEx"(lpRect As RECT, ByVal dsStyle As Long, ByVal bMenu As Long, ByVal dwEsStyle AsLong) As Long

Công dụng: Điều chỉnh cửa sổ khi có vùng làm việc client (Không tính kích thước củathanh tiêu đề, đường viền và các phần thêm) được khai báo, khi biết kiểu cửa sổ.Tham số kèm: LpRect Hình chữ nhật chứa vùng làm việc client. DwStyle Kiểu cửa sổ.BMenu Đưa giá trị True (Khác zero) nếu cửa sổ có trình đơn DwEsStyle kiểu cửa sổ mởrộng.

## 1.2 Tổng quan về tiến trình

### 1.2.1 Tiến trình là gì?

Tiến trình là một thực thể đang thực hiện điều khiển một đoạn mã lệnh riêng không gian,địa chỉ, ngăn xếp và sở hữu một trạng thái giúp thông báo nó đang làm gì (đang chạy,đang chờ, đã đóng, ...).Tiến trình có 4 thành phần quan trọng: CPU, bộ nhớ, File, Thiết bị nhập xuất.

### 1.2.2 Chương trình ví dụ về tiến trình

- Khai báo hiển thị danh sách các tiến trình

private void loadprocessList()

{

listView1.Items.Clear();

Process[] processList = Process.GetProcesses();

foreach (Process process in processList)

{

ListViewItem item = new ListViewItem(process.ProcessName);

listView1.Items.Add(item); );// Hiển thị danh sách tiến trình dạng list

}

}

* Xử lý sự kiện mở 1 tiến trình

private void btnStart\_Click(object sender, EventArgs e)

{

string text = textBox1.Text;

Process process = new Process();

process.StartInfo.FileName = text;

process.Start();

loadprocessList();

}

* Xử lý sự kiện dừng 1 tiến trình

private void btnStop\_Click(object sender, EventArgs e)

{

ListViewItem item = listView1.SelectedItems[0];

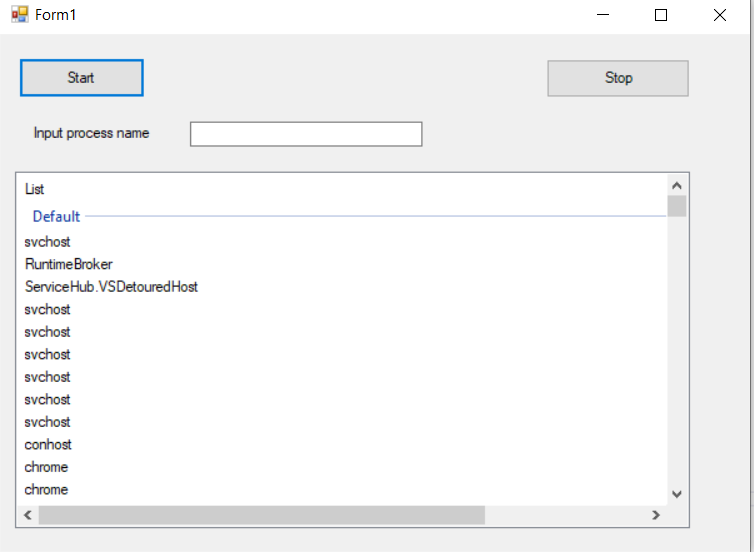
Process process =(Process) item.Tag;

process.Kill();

loadprocessList();

}

* Hình ảnh chạy chương trình



# Chương 2: Chương trình ứng dụng hẹn giờ tự tắt máy tính và tự động tắt máy khi hết pin

## 2.1 Giới thiệu bài toán và chương trình ứng dụng

* Chương trình hẹn giờ tự động tắt máy tính cho phép máy tính tự động tắt khi người dùng k có bất kỳ thao tác nào như di chuột, gõ phím hay cảm ứng trên màn hình ( đối với những máy có màn hình cảm ứng)
* chương trình sẽ sử dụng hỗ trợ hàm API trên hệ điều hành window tiến trình của chương trình sẽ chạy ngầm định cho phép máy tính tự động tắt máy khi đạt đến thời gian chờ đã quy định trước

## 2.2 Mã nguồn chương trình ứng dụng và thực nghiệm

### 2.2.1 Mã nguồn chương trình

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace BTL\_HDH\_

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

LoadStatusStripHuy();

}

// khởi tạo biến

decimal downtime = 0;

string statustext = "";

PowerStatus status = SystemInformation.PowerStatus;

decimal battery;

decimal Batterypercentageoff;

// Hàm LoadStatusStrip hiển thị thời gian đếm ngược

void LoadStatusStrip()

{

var item = new ToolStripLabel() { Text = statustext};

statusStrip1.Items.Add(item);

var item1 = new ToolStripLabel() { Text = downtime.ToString() };

statusStrip1.Items.Add(item1);

}

// hàm LoadStatusStripHuy xóa thời gian đếm ngược

void LoadStatusStripHuy()

{

var item = new ToolStripLabel() { Text = " " };

statusStrip1.Items.Add(item);

}

// tạo sự kiện nmGiay\_ValueChanged khi biến của giây tăng lên 60 tăng phút lên 1 và giây về 0

private void nmGiay\_ValueChanged(object sender, EventArgs e)

{

NumericUpDown box = sender as NumericUpDown;

if (box.Value >= 60)

{

nmPhut.Value++;

box.Value = 0;

}

}

// tạo sự kiện nmPhut\_ValueChanged khi biến của phút tăng lên 60 tăng giờ lên 1 và phút về 0

private void nmPhut\_ValueChanged(object sender, EventArgs e)

{

NumericUpDown box = sender as NumericUpDown;

if (box.Value >= 60)

{

nmGio.Value++;

box.Value = 0;

}

}

// Hàm CalculateDowntime tính thời gian chuyển về giây

void CalculateDowntime()

{

downtime = nmGiay.Value + nmPhut.Value \* 60 + nmGio.Value \* 60 \* 60;

}

//shutdown theo comand

void ShutDown(string command)

{

System.Diagnostics.Process.Start("shutdown", command); // gọi hàm api shutdown của Windows

}

// hàm xử lý hiển thị thời gian đếm ngược ở statusStrip

private void Timeturnoff\_Tick(object sender, EventArgs e)

{

downtime--;

statusStrip1.Items.Clear();

LoadStatusStrip();

}

// Hàm xử lý nút Shutdown

private void btShotdow\_Click(object sender, EventArgs e)

{

statustext = "Máy sẽ shutdow trong: ";

LoadStatusStrip();

CalculateDowntime();

ShutDown("-s -t " + downtime.ToString()); // gọi hàm shutdown với time = downtime

Timeturnoff.Start();

}

// Hàm xử lý nút Restart

private void btRestart\_Click(object sender, EventArgs e)

{

statustext = "Máy sẽ restart trong: ";

LoadStatusStrip();

CalculateDowntime();

ShutDown("-r -t " + downtime.ToString()); // gọi hàm shutdown với time = downtime

Timeturnoff.Start();

}

// Hàm xử lý nút Cancel

private void btCancel\_Click(object sender, EventArgs e)

{

statusStrip1.Items.Clear();

LoadStatusStripHuy();

ShutDown("-a");

Timeturnoff.Stop();

}

// Hàm xử lý hiển thị phần trăm pin hiện tại

private void realtime\_Tick(object sender, EventArgs e)

{

// Gọi hàm api của Windows lấy phần trăm pin hiện tại chuyển về chuỗi và gán cho Text Charge

txtCharge.Text = status.BatteryLifePercent.ToString("P0");

battery = Convert.ToDecimal(status.BatteryLifePercent) \* 100;

NMR\_shutdowwithbattery.Maximum = battery;

NMR\_shutdowwithbattery.Minimum = 5;

}

private void Form1\_Load(object sender, EventArgs e)

{

realtime.Start();

realtime.Enabled = true;

}

// Hàm xử lý nút Bật tự động tắt

private void BT\_TurnOn\_Click(object sender, EventArgs e)

{

BT\_TurnOn.Enabled = false; // không cho phép bấm vào button khi đã thực hiện bấm 1 lần

NMR\_shutdowwithbattery.Enabled = false;// không cho phép thay đổi phần trăm pin khi đã bấm vào nút khởi động

checkturnoff.Start();

}

// Hàm xử lý nút Hủy

private void BT\_Cancel\_Click(object sender, EventArgs e)

{

NMR\_shutdowwithbattery.Enabled = true;

BT\_TurnOn.Enabled = true;

checkturnoff.Stop();

ShutDown("-a");

}

// Hàm xử lý tắt máy tự động khi hết pin

private void checkturnoff\_Tick(object sender, EventArgs e)

{

Batterypercentageoff = Convert.ToInt32(NMR\_shutdowwithbattery.Value);

if (Batterypercentageoff == battery)

{

ShutDown("-s -t " + 30); // gọi hàm shutdown với time 30s

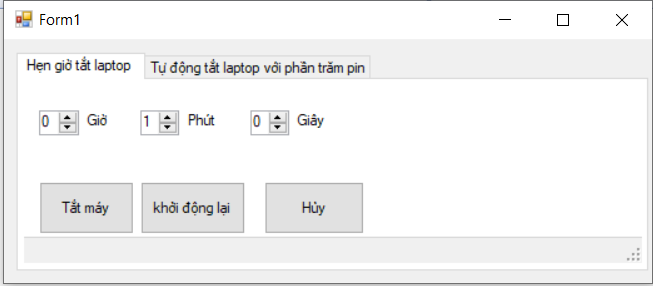
}

}

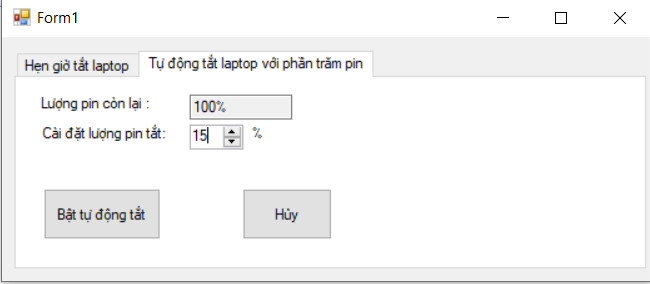
}

}

### 2.2.2 Hình ảnh thực nghiệm chương trình



Hình 1: Giao diện hẹn giờ tắt, restart lại máy



Hình 2: Giao diện tự động tắt laptop và phần trăm pin

# Chương 3: Kết luận và hướng phát triển

## 3.1 Kết luận.

Tính năng tự động tắt máy ở máy tính là 1 tính năng cần thiết cần được trang bị bởi tính đặc thù của nó. Tự động tắt máy sẽ giúp tiết kiệm tài nguyên, thuận tiện cho người dùng khi mà có việc cần phải xử lý gấp mà k thể làm liên tiếp các thao tác liên tục.

Sau thời gian nghiên cứu và học hỏi nhóm chúng em đã xây dựng 1 ứng dụng cơ bản để hẹn giờ tự động tắt máy tính. Tuy nhiên chương trình của chúng em còn khá đơn giản và nhiều sai sót rất mong nhận được sự góp ý từ thầy. Nhóm em xin chân thành cảm ơn

## 3.2 Tài liệu tham khảo

* Bài giảng môn Hệ điều hành
* <https://www.youtube.com/watch?fbclid=IwAR1jy2Ye5nkDUsPFzZaswSQqOf_gJ4hblczyKx9QsPd5c_FjEXDzyrohCYs&v=eHqIMyg07b0&feature=youtu.be>
* Tài liệu tham khảo về hàm API