객체지향윈도우즈

프로그래밍

프로젝트 보고서

2018.12.12

컴퓨터공학과

2015111489 김성현

오세만 교수님

**1.프로그램에 들어가는 기능 정리**

**- 절전 모드 기능과 자동 종료 기능을 하는 프로그램 설계**

* 주의 사항

1. 해당 프로그램은 노트북 실행을 바탕으로 만들어졌다.
2. nircmd파일을 실행할 시에, 코드상에서 nircmd파일의 위치 주소를 바꾸어 줘야 제대로 실행할 수 있다.
3. 컴퓨터 관리 part
4. 모니터 밝기 조절 기능

모니터의 밝기를 버튼이나 스크롤 바를 이용해서 조절할 수 있다.

1. 컴퓨터 볼륨 조절 기능

컴퓨터 볼륨을 버튼이나 스크롤 바를 이용해서 조절할 수 있다.

1. 배터리 정보 열람 기능

배터리의 현재 값에 대해서 얼마나 남았는지, 충전 여부등과 같은 정보를 사용자가 볼 수 있다.

1. 파워 관리 part
2. 절전모드

최소 절전이나 최대 절전모드를 바로 실행할 수 있다. 또한 최소 절전과 최대 절전을 사용자가 컴퓨터에 아무 반응을 보이지 않을 시 몇 분까지 기다렸다가 절전모드에 들어갈 것인지 저장하고 수정할 수 있다.

1. 자동종료

프로세스가 없을 때만 종료하는 정상종료와 프로세스의 여부와 관계없이 컴퓨터를 종료하는 강제 종료를 시간에 맞춰서 조절할 수 있다.

1. 단축키를 이용한 절전모드 실행
2. 윈도우 바에 나타나는 메뉴 창을 이용해서 단축키를 목록 중에서 단축키를 저장해서 절전모드를 실행할 수 있다.

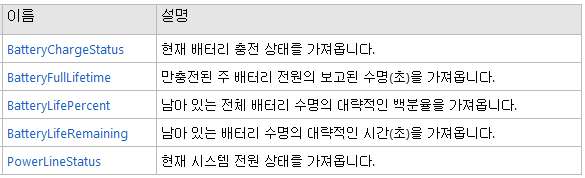
**2.프로그램 설계와 구현 기술 방법**

1. 컴퓨터 관리 part
2. 모니터 밝기 조절 기능

Trackbar을 이용하고, trackbar의 스크롤이 움직일때마다 nircmd를 통해서 모니터의 밝기를 +10, -10로 조절한다.

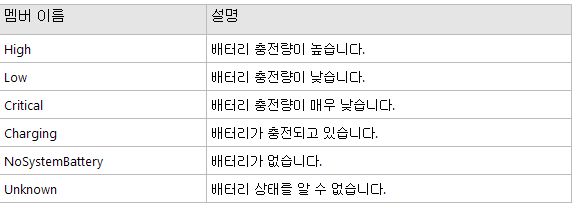
1. 컴퓨터 볼륨 조절 기능

Trackbar을 이용하고, trackbar의 스크롤이 움직일 때마다 nircmd를 통해서 컴퓨터의 볼륨 크기를 전체 볼륨 max 크기의 1/10으로 줄이거나 증가하는 등의 조절을 한다.

1. 배터리 정보 열람 기능

배터리에 대한 값을 가져오기 위해서 PowerStatus 클래스를 사용한다. Timer을 추가해서 timer가 tick을 하는 interval시간마다 새로운 배터리에 상태에 대한 값들을 호출한다. 호출한 값에 따라서 배터리에 대한 그림과 출력을 다르게 시킨다. 아래의 표는 PowerStatus 클래스로부터 값을 받아올 수 있는 함수의 종류다.

아래의 표는 BatteyChageStatus의 멤버 이름이다.



PowerLineStatus를 이용해서 해당 컴퓨터가 충전중인지 여부또한 확인할 수 있다.

1. 파워 관리 part
2. 절전모드

최소 절전 라디오 버튼을 누른 뒤 바로 실행을 누르면 최소 절전이 실행된다. 이는 nircmd로 기능을 하였다.

최대 절전 라디오 버튼을 누른 뒤 바로 실행을 누르면 최대 절전이 실행된다. 이는 역시 nircmd를 이용하였다.

combobox에서 시간을 선택하고, 절전모드를 선택한 뒤 저장하면 특정 시간 뒤에 절전모드가 시행되는지 사용자가 직접 선택할 수 있다. 이때는, nircmd가 아니라 powercfg.dll을 이용하였다. cmd창을 불러와서 cmd창에서 “powercfg /x standby-timeout-dc 시간” 과 같이 명렁어를 주면 실행하게 되므로, c#에서 cmd창을 불러와서 실행시켰다.

1. 자동종료

정상 종료와 강제 종료로 두가지로 나누어져 있으며, 정상 종료나 강제 종료를 누르게 되면 timer을 이용해서 계속해서 시간을 받아오다가 정상 종료나 강제종료를 저장한 시간과 같은 시간이 되었을 때 정상종료나 강제 종료를 실행하게 된다. 만일 ‘사용 안함’ 버튼을 누르면 저장 해 둔 정상 종료와 강제 종료가 있다면 해당 정보를 삭제하고 자동종료가 실행되지 않게 된다.

1. 단축키를 이용한 절전모드 실행

윈도우 바에서 절전 모드 단축키 설정 메뉴를 누르게 되면 단축키를 등록할 수 있으며, 단축키를 등록하는 폼은 form4이다. Form4에 단축키에 대한 번호를 저장할 수 있는 프로퍼티를 생성하여 단축키의 번호를 저장하게 되면 , form1로 넘겨 단축키를 생성하게 된다. 만일 이 프로그램이 윈도우 바를 이용해서 종료가 되게 되면 단축키가 해제되게 된다.

**3.소스코드/주석**

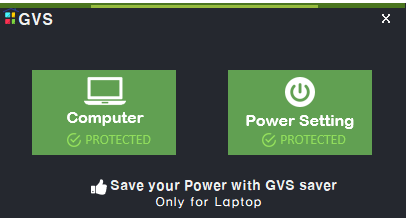
|  |
| --- |
| Form1.cs |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.IO;  using HTTPComm;  using System.Diagnostics;  using System.Runtime.InteropServices;  namespace HTTPCommTester  {  public partial class Form1 : Form  {  Point fPt;//마우스 커서의 point  bool isMove;//마우스가 움직인지에 대한 여부  int standby\_key = 0;//단축키 값을 받을 때, 단축키 설정 값을 form4로부터 받아와 저장하기 위한 변수  HTTPWebComm comm = new HTTPWebComm();  public Form1()  {  InitializeComponent();  //이미지 불러옴  pictureBox1.Image = Properties.Resources.computer;  pictureBox2.Image = Properties.Resources.powerSetting;  pictureBox3.Image = Properties.Resources.thumbs\_up;  pictureBox4.Image = Properties.Resources.applications;  //notifyicon 실행  Tray\_Icon();  }  private void Tray\_Icon()  {  //notifyicon을 contexmenustrcip에 연결하고 윈도우 바에 노출  notifyIcon1.ContextMenuStrip = contextMenuStrip1;  textBox1.ContextMenuStrip = contextMenuStrip1;  notifyIcon1.Visible = true;  ShowInTaskbar = true;  WindowState = FormWindowState.Normal;  }  //form의 스타일을 none으로 지정했기 때문에 panel을 배경에 깔아두고 마우스로 form움직임을 실행하기 위한 함수들  private void panel1\_MouseDown(object sender, MouseEventArgs e)  {  isMove = true;  fPt = new Point(e.X, e.Y);  }  private void panel1\_MouseUp(object sender, MouseEventArgs e)  {  isMove = false;  }  private void panel1\_MouseMove(object sender, MouseEventArgs e)  {  if (isMove && (e.Button & MouseButtons.Left) == MouseButtons.Left)  Location = new Point(this.Left - (fPt.X - e.X), this.Top - (fPt.Y - e.Y));  }  //단축키  [DllImport("user32.dll")]  public static extern bool RegisterHotKey(IntPtr itp, int id, KeyInform fsInform, Keys vk);  [DllImport("user32.dll")]  public static extern bool UnregisterHotKey(IntPtr itp, int id);  const int KEYid = 31197;  public enum KeyInform  {  None = 0,  Alt = 1,  Control = 2,  Shift = 4,  Windows = 8  }  const int HOTKEYGap = 0x0312;  protected override void WndProc(ref Message message)  {  switch (message.Msg)  {  case HOTKEYGap:  Keys key = (Keys)(((int)message.LParam >> 16) & 0xFFFF); //눌러 진 단축키의 키  KeyInform modifier = (KeyInform)((int)message.LParam & 0xFFFF);//눌려진 단축키의 수식어  if ((KeyInform.Control) == modifier && Keys.Q == key && standby\_key == 0)//단축키 설정에서 q로 저장했을때  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=sleep");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=sleep");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  this.절전모드실행ToolStripMenuItem.PerformClick();  }  else if ((KeyInform.Control) == modifier && Keys.W == key && standby\_key == 1)//단축키 설정에서 w로 저장했을때  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=sleep");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=sleep");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  this.절전모드실행ToolStripMenuItem.PerformClick();    }  break;  }  base.WndProc(ref message);  }  private void Form1\_FormClosed(object sender, FormClosedEventArgs e)//폼이 닫힐때  {  UnregisterHotKey(this.Handle, KEYid);//단축키 제거  }  private void button2\_Click(object sender, EventArgs e)//x버튼이 눌렸을때 폼 종료 //그러나 윈도우 바에는 나타냄  {  this.Close();  }  private void pictureBox1\_Click(object sender, EventArgs e)//그림 1을 눌럿을때 컴퓨터 관리 form 실행  {  Form2 form2 = new Form2();  form2.Show();  }  private void pictureBox2\_Click(object sender, EventArgs e)//그림 2를 눌렀을때 파워 관리 form 실행  {  Form3 form3 = new Form3();  form3.Show();  }  private void 종료ToolStripMenuItem\_Click(object sender, EventArgs e)//종료 contexstripmenu 클릭시  {  if (MessageBox.Show("종료하시겠습니까?\n종료시 자동종료 내용이 삭제됩니다.", "Exit GVS program", MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)  {  this.notifyIcon1.Visible = false;//notifyicon 안보이게 하기  UnregisterHotKey(this.Handle, KEYid);//단축키 제거  Application.Exit();//실행 종료  }  }  private void notifyIcon1\_MouseDoubleClick(object sender, MouseEventArgs e)  {  this.Visible = true; //notifyicon 더블 클릭시에 해당 어플리케이션을 보여줌  if (this.WindowState == FormWindowState.Minimized)  this.WindowState = FormWindowState.Normal; // 최소화를 멈ㅊ ㅁ  this.Activate(); // 폼을 활성화  }  private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  if (e.CloseReason == CloseReason.UserClosing) //종료하려 할떄  {  e.Cancel = true;//종료를 취소  this.Visible = false;//어플리케이션을 숨김  }  }  private void 절전모드단축키설정ToolStripMenuItem\_Click\_1(object sender, EventArgs e)  {  Form4 frm4 = new Form4();//form4 실행  frm4.ShowDialog();  standby\_key = frm4.Set\_standby\_key;//form4로부터 받아온 값을 저장  if(standby\_key ==0)//form4로부터 받아온 값에 따라서 다른 단축키 추가  {  RegisterHotKey(this.Handle, KEYid, KeyInform.Control, Keys.Q); //CTRL+Q로 단축키 추가  }  else if(standby\_key ==1)  {  RegisterHotKey(this.Handle, KEYid, KeyInform.Control, Keys.W); //CTRL+Wㅗ 단축키 추가  }  }  private void 절전모드실행ToolStripMenuItem\_Click\_1(object sender, EventArgs e)  {  //절전모드 실행, 서버에 log 전송    //nircmd를 이용해서 standby(절전) 실행  //nircmd의 위치가 local이기 때문에 다른사람이 실행시에는 위치를 변경해야햄  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "standby");  }  private void textBox1\_TextChanged(object sender, EventArgs e)  {  }  }  } |

|  |
| --- |
| Form2.cs |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.Runtime.InteropServices;  using System.Diagnostics;  using HTTPComm;  namespace HTTPCommTester  {  public partial class Form2 : Form  {  //마우스 관련 변수 // form1과 같으므로 생략  Point fPt;  bool isMove;  //화면 밝기와 소리 관련 변수  int trackbar1\_count = 0;  int trackbar1\_line = 5;  int trackbar2\_count = 0;  int trackbar2\_line = 5;  HTTPWebComm comm = new HTTPWebComm();  private Timer MonitorTimer = new Timer();//timer 선언  public Form2()  {  InitializeComponent();  pictureBox4.Image = Properties.Resources.applications;  //사용자의 기본적인 세팅을 다음과 같이 밝기 50, 소리 32767로 설정  //local이므로, nircmd의 위치는 실행하는 사람에 따라 바뀌어야 함  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "setbrightness 60");  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "setsysvolume 32767");  //밝기와 소리의 기본 세팅에 대한 trackbar의 기준 값  trackbar1\_line = 5;  trackbar2\_line = 5;  label15.BackColor = Color.Transparent;  }  protected override void OnLoad(EventArgs e)  {  //배터리에 대한 정보값을 받아와서 업데이트 하기 위해서 timer를 실행  base.OnLoad(e);  this.MonitorTimer.Interval = 2000; // 2초에 한번씩 상태를 체크  this.MonitorTimer.Tick += new EventHandler(MonitorTimer\_Tick);//interval만큼 시간이 지날때마다 해당 기능도 실행  this.MonitorTimer.Start();  // 최초 호출  this.UpdateText();  this.UpdateIcon();  }  protected override void OnClosing(CancelEventArgs e)//종료시 닫고, 타이머도 종료  {  base.OnClosing(e);  this.MonitorTimer.Stop();  }  void MonitorTimer\_Tick(object sender, EventArgs e)  {  this.UpdateText();  this.UpdateIcon();  }  private void UpdateText() //배터리에 대한 정보를 업데이트하는 함수  {  PowerStatus status = SystemInformation.PowerStatus;  //노트북의 배터리 용량에 대한 값  if(status.BatteryChargeStatus.ToString() == "High") { this.label11.Text = "배터리 충전량이 높습니다."; }  else if(status.BatteryChargeStatus.ToString() =="Low") { this.label11.Text = "배터리 충전량이 낮습니다."; }  else if (status.BatteryChargeStatus.ToString() =="Critical") { this.label11.Text = "배터리 충전량이 매우 낮습니다."; }  else { this.label11.Text = "-"; }    //노트북이 충전기와 연결되었는지에 대한 여부  if (status.PowerLineStatus.ToString() == "Offline") { this.label13.Text = "-"; }  else if (status.PowerLineStatus.ToString() == "Online") { this.label13.Text = "케이블로 충전중"; }  else { this.label13.Text = " - "; }  //그외 배터리에 대한 정보들을 받아와 저장하여 출력  this.label14.Text = "현재 남은 배터리 시간 : " + (status.BatteryLifeRemaining != -1 ? TimeSpan.FromSeconds(status.BatteryLifeRemaining).ToString() : "-");  this.progressBar1.Value = (int)(status.BatteryLifePercent != 255 ? status.BatteryLifePercent \* 100 : 0);  this.label15.Text = this.progressBar1.Value.ToString() + " % ";  }  private void UpdateIcon()// 받아온 값에 따라서 그림이 달라질 수 있도록 하는 함수  {  PowerStatus status = SystemInformation.PowerStatus;    if (status.PowerLineStatus == PowerLineStatus.Offline)//충전중이 아닐때 0.x는 노트북의 배터리가 (10\*x)%임을 의미  {  //남은 배터리 양에 따라서 다른 그림들을 출력  if (status.BatteryLifePercent < 0.2)  {  pictureBox1.Image = Properties.Resources.battery\_\_12;  }  else if (status.BatteryLifePercent < 0.4 && status.BatteryLifePercent >= 0.2)  {  pictureBox1.Image = Properties.Resources.battery\_\_11;  }  else if (status.BatteryLifePercent < 0.6 && status.BatteryLifePercent >= 0.4)  {  pictureBox1.Image = Properties.Resources.battery\_\_10;  }  else if (status.BatteryLifePercent < 0.8 && status.BatteryLifePercent >= 0.6)  {  pictureBox1.Image = Properties.Resources.battery\_\_9;  }  else  {  pictureBox1.Image = Properties.Resources.battery\_\_8;  }  }  else if (status.PowerLineStatus == PowerLineStatus.Online)//케이블로 충전중일때  {  pictureBox1.Image = Properties.Resources.battery\_\_13;  }  else if (status.PowerLineStatus == PowerLineStatus.Unknown)//알수없는 상태일때  {  pictureBox1.Image = Properties.Resources.info\_\_1;  }    }  //마우스 관련 함수들 // form1과 같으므로 생략  private void panel1\_Mousedown(object sender, MouseEventArgs e)  {  isMove = true;  fPt = new Point(e.X, e.Y);  }  private void panel1\_Mouseup(object sender, MouseEventArgs e)  {  isMove = false;  }  private void panel1\_Mousemove(object sender, MouseEventArgs e)  {  if (isMove && (e.Button & MouseButtons.Left) == MouseButtons.Left)  Location = new Point(this.Left - (fPt.X - e.X), this.Top - (fPt.Y - e.Y));  }  //밝기를 조절하려고 trackbar를 조절했을때 실행하는 함수  private void trackBar1\_Scroll(object sender, EventArgs e)  {    trackbar1\_count = trackBar1.Value;//현재 trackbar의 값 받아옴  if (trackbar1\_count < trackbar1\_line) // 원래보다 작은쪽으로 스크롤 했을 때  {  //밝기값 10 감소  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "changebrightness -10");  }  else //원래 기준보다 큰쪽으로 스크롤 했을 때  {  //밝기값 10 증가  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "changebrightness +10");  }  //원래 초기화된 값이 5였는데, 사용자가 임의로 줄이거나 늘였기 때문에 기준값을 새로 저장  trackbar1\_line = trackbar1\_count;  }  //소리를 조절하려고 trackbar를 조절했을때 실행하는 함수  private void trackBar2\_Scroll(object sender, EventArgs e)  {  trackbar2\_count = trackBar2.Value;//현재 trackbar의 값 받아옴  if (trackbar2\_count < trackbar2\_line) // 원래보다 작은쪽으로 스크롤 했을 때  {  //max볼륨의 1/10만큼을 감소  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "changesysvolume -6554");  }  else //원래 기준보다 큰쪽으로 스크롤 했을 때  {  //max볼륨의 1/10만큼 증가  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "changesysvolume +6554");  }  //원래 초기화된 값이 중간값 이였는데, 사용자가 임의로 줄이거나 늘였기 때문에 기준값을 새로 저장  trackbar2\_line = trackbar2\_count;  }  private void button2\_Click\_1(object sender, EventArgs e)//x버튼을 눌렀을때 해당 form 종료  {  this.Close();  }  //mute버튼 클릭시  private void button3\_Click(object sender, EventArgs e)  {  trackbar2\_count = trackBar2.Value;  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "mutesysvolume 1");  trackBar2.Value = 0;//trackbar의 값 0으로 저장  }  //max volume 버튼을 눌럿을 때  private void button4\_Click(object sender, EventArgs e)  {  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "setsysvolume 65535");  //소리관련 변수들 max값인 10으로 저장  trackBar2.Value = 10;  trackbar2\_count = 10;  trackbar2\_line = 10;  }  //unmute버튼을 눌렀을 때  private void button5\_Click(object sender, EventArgs e)  {  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "mutesysvolume 0");  trackBar2.Value = trackbar2\_count;  }  }  } |

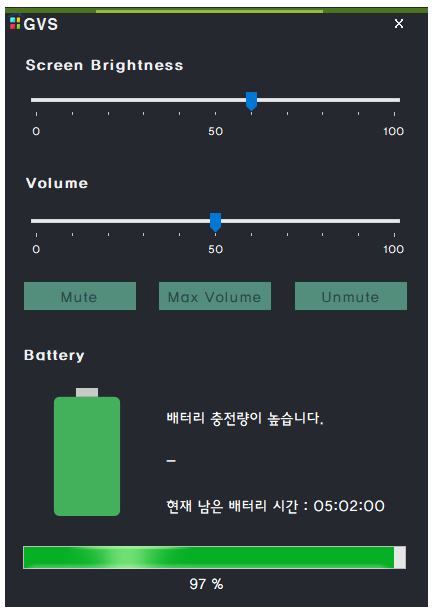
|  |
| --- |
| Form3.cs |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Diagnostics;  using System.Runtime.InteropServices;  using System.Windows;  using HTTPComm;  namespace HTTPCommTester  {  public partial class Form3 : Form  {  //마우스 관련 변수들  Point fPt;  bool isMove;  String temp;  int hour = DateTime.Now.Hour;//현재시간의 시를 저장  int min = DateTime.Now.Minute;//현재시간의 분을 저장  int sec = DateTime.Now.Second;  int hour2 = DateTime.Now.Hour;  int min2 = DateTime.Now.Minute;  int sec2 = DateTime.Now.Second;  string datePart;    int com\_off\_force = 0;//정상종료에 대한 flag  int com\_off\_stand = 0;//강제종료에 대한 flag  int com\_standby = 0;  int com\_hibernate = 0;  HTTPWebComm comm = new HTTPWebComm();  public Form3()  {  InitializeComponent();  //그림들 불러옴  pictureBox4.Image = Properties.Resources.applications;  pictureBox1.Image = Properties.Resources.border;  pictureBox2.Image = Properties.Resources.border;  //comboBox의 기본값들 설정  comboBox1.Text = "사용안함";  comboBox1.SelectedText = "사용안함";  comboBox2.Text = "1분";  comboBox2.SelectedText = "1분";  //timer의 기본 값 설정 1초마다.  timer1.Interval = 1000;  timer1.Start();  label8.Visible = false;  }  // timer의 interval만큼마다 실행하는 이벤트를 생성  private void timer1\_Tick(object sender, EventArgs e)  {  //현재시간 불러와서 출력  DateTime dt = DateTime.Now;  datePart = dt.ToString("yyyy년MM월dd일hh:mm:ss");  label6.Text = "현재 시간 : " + datePart;  //자동종료나 강제종료 관련된 flag가 들어와 있는 경우 몇시에 작동을 할건지 출력하기 위한 조건문  if (com\_off\_stand == 1)//정상종료  {  label8.Visible = true;  label8.Text = hour + "시 " + min + "분 " + sec +"초에 정상 자동 종료를 실행\n";  }  else if (com\_off\_force == 1)//강제종료  {  label8.Visible = true;  label8.Text = hour + "시 " + min + "분 " + sec + "초에 강제 자동 종료를 실행\n";  }  else if (com\_off\_stand == 0 || com\_off\_force == 0)//자동종료가 설정 안된 경우 label을 숨김  label8.Visible = false;  if(com\_standby ==1)  {  if (dt.Hour == hour2 && dt.Minute == min2 && dt.Second == sec2)  {  Standby();  com\_standby = 0;  }  }  else if (com\_hibernate == 1)  {  if (dt.Hour == hour2 && dt.Minute == min2 && dt.Second ==sec2)  {  Hibernate();  com\_hibernate = 0;  }  }  if (com\_off\_stand == 1)//정상종료 플래그가 들어와있을때  {  //자동종료로 지정한 값과 현재 시간이 같을 때  if (dt.Hour == hour && dt.Minute == min && dt.Second == sec)  {  com\_off\_stand = 0;//플래그 수정  Com\_standby();  }  }  else if(com\_off\_force ==1)//강제종료  {  //지정 시간과 현재 시간이 같은 경우에  if (dt.Hour == hour && dt.Minute == min && dt.Second == sec)  {  com\_off\_force = 0;//플래그 0으로 저장  Com\_force();  }    }    }  //마우스 관련 이벤트 함수들  private void panel1\_MouseDown(object sender, MouseEventArgs e)  {  isMove = true;  fPt = new Point(e.X, e.Y);  }  private void panel1\_Mouseup(object sender, MouseEventArgs e)  {  isMove = false;  }  private void panel1\_Mousemove(object sender, MouseEventArgs e)  {  if (isMove && (e.Button & MouseButtons.Left) == MouseButtons.Left)  Location = new Point(this.Left - (fPt.X - e.X), this.Top - (fPt.Y - e.Y));  }  //종료 모드에 대해서 저장 버튼을 눌렀을 때  private void button1\_Click(object sender, EventArgs e)  {  //라디오 버튼에 대해서 아무것도 적히지 않은채 저장 버튼이 눌렸을 때  if (!(radioButton2.Checked) && !(radioButton4.Checked) && !(radioButton5.Checked))  {  MessageBox.Show("종료 모드를 선택해주세요.");//메세지창 출력  }  else if (radioButton2.Checked)//사용안함 버튼 선택시  {  //자동 종료 관련 플래그들 값을 0으로 지정  com\_off\_stand = 0;  com\_off\_force = 0;  }  else//정상 종료나 강제 종료 버튼이 눌러진 상태로 저장을 누른 경우  {  if (((com\_off\_force == 1) || (com\_off\_stand == 1)))//현재 자동종료가 설정되어 있는 경우가 아닐때만 실행 //둘중 하나라도 1일떄  {  MessageBox.Show("이미 자동 종료모드가 실행중입니다. \n 모드를 변경하려면 사용안함을 저장 한 후, 새로 저장해주세요.");  }  else//현재 자동종료가 설정되어 있는 경우가 아닐때만 실행  {  //현재시간의 시와 분을 변수에 따로 각자 저장  hour = DateTime.Now.Hour;  min = DateTime.Now.Minute;  sec = DateTime.Now.Second;  if (comboBox2.Text != "1분")//시간으로 저장되어 있는 것들  {  //temp에 선택한 시간에 대해서 숫자만 받아와서 hour변수에 저장  temp = comboBox2.Text.Substring(0, comboBox2.Text.IndexOf("시간"));  String temp2 = hour + temp;  hour = Convert.ToInt32(temp) + hour;  //24시부터는 -24를 해서 0시로 만듦  if (hour >= 24) { hour -= 24; }  }  else if (comboBox2.Text == "1분")//첫번째 값인 30분일때.  {  min += 1;//분에 해당 분을 더해줌  if (min >= 60)//더한값이 60분일때.  {  min -= 60;//60을 빼고  hour += 1;//시간을 1시간 더해줌  if (hour == 24) hour = 0;//더한 시간이 24시가 되면 0으로 지정  }  }  if (radioButton4.Checked) //정상종료 버튼이 눌린경우  {  com\_off\_stand = 1;//정상종료 플래그를 1로 저장  }  else if (radioButton5.Checked)//강제종료 버튼이눌린경우  {  com\_off\_force = 1;//강제종료 플래그를 1로 저장  }  }  }  }  private void button2\_Click(object sender, EventArgs e)//x버튼을 눌렀을 때  {  this.Visible = false;//form 숨김  }  private void Standby()  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=sleep");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=sleep");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "standby");//nircmd를 이용해서 절전모드 바로 실행  }  private void Com\_standby()//로그를 보내고 정상종료 실행  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=shutdown");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=shutdown");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  //현재 실행중인 프로세스의 모곡을 받아와 배열에 저장  Process[] pro = Process.GetProcesses();  //프로세스가 켜져있는게 존재하게 되면  if (pro.Length > 0)  {  //메세지 창을 띄움  MessageBox.Show("프로세스가 실행중이기 때문에, 자동종료를 거부합니다.");  }  else if (pro.Length == 0)  {// 켜져있는 프로세스가 없으면 nircmd를 이용해서 컴퓨터 종료  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "exitwin poweroff ");  }  }  private void Com\_force() //log를 보내고 강제종료실행  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=shutdown");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=shutdown");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  com\_off\_force = 0;//플래그 0으로 저장  Process.Start("C:\\Users\\gkzns\\OneDrive\\바탕 화면\\nircmd\\nircmd.exe", "exitwin poweroff ");//nircmd를 이용해서 강제종료  }  private void Hibernate()  {  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=hibernate");  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=read&action=hibernate");  comm.setMessage("");  comm.Reqeust();  string result = comm.Response();  textBox1.Text = result;  Process.Start(fileName: "rundll32", arguments: "powrprof.dll, SetSuspendState");//nircmd를 이용해서 최대절전모드 바로 실행  }  private void button3\_Click(object sender, EventArgs e)//절전모드 바로실행 버튼을 눌렀을때 실행되는 이벤트  {  //절전모드를 선택하는 라디오 버튼 아무것도 선택하지 않고 바로실행 버튼을 눌렀을 대  if (!(radioButton1.Checked) && !(radioButton3.Checked))  {  MessageBox.Show("절전모드를 선택해 주세요.");//메세지 박스 출력  }  else if (radioButton1.Checked)//최소절전 선택시  {  Standby();  }  else if(radioButton3.Checked)//최대절전 선택시  {  Hibernate();  }  }  private void button4\_Click(object sender, EventArgs e)//절전모드에서 저장 버튼을 눌렀을 때  {  if (!(radioButton1.Checked) && !(radioButton3.Checked))//절전모드 선택안하고 저장 버튼 클릭시  {  MessageBox.Show("절전모드를 선택해 주세요.");//메세지 박스 출력  }  else  {  hour2 = DateTime.Now.Hour;  min2 = DateTime.Now.Minute;  sec2 = DateTime.Now.Second;  if (comboBox1.Text != "사용안함" && radioButton1.Checked)//최소절전  {  if (com\_hibernate == 1) com\_hibernate = 0;  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=sleep");  String msg = comboBox1.Text.Substring(0, comboBox1.Text.IndexOf("분"));  min2 += Convert.ToInt32(msg);  if (min2 >= 60)  {  min2 -= 60;  hour2++;  if (hour2 == 24) hour2 -= 24;  }  com\_standby = 1;  }  else if (comboBox1.Text != "사용안함" && radioButton3.Checked)//최대절전  {  if (com\_standby == 1) com\_standby = 0;  comm.SetURL("http://210.94.194.82:52131/log.asp?id=2015111489&cmd=write&action=hibernate");  String msg = comboBox1.Text.Substring(0, comboBox1.Text.IndexOf("분"));  min2 += Convert.ToInt32(msg);  if (min2 >= 60)  {  min2 -= 60;  hour2++;  if (hour2 == 24) hour2 -= 24;  }  com\_hibernate = 1;  }  else if (comboBox1.Text == "사용안함" && radioButton1.Checked)//최소절전을 사용안함으로 저장하면  {  com\_standby = 0;  }  else if (comboBox1.Text == "사용안함" && radioButton3.Checked)//최대절전을 사용안함으로 저장하면  {  com\_hibernate = 0;  }  }  }  private void label5\_Click(object sender, EventArgs e)//절전모드 도움말 클릭시  {  MessageBox.Show("최소 절전은 마우스와 키보드 키 등으로 절전모드를 종료할 수 있습니다.\n\n최대 절전은 power on 버튼을 눌러 절전모드를 종료할 수 있습니다.");  }  private void label1\_Click(object sender, EventArgs e)//자동종료 도움말 클릭시  {  MessageBox.Show("자동 종료는 어떤 파일이나 문서가 켜져있으면 종료되지 않습니다.\n\n강제 종료는 어떤 문서나 파일이 켜져있어도 컴퓨터가 종료됩니다.");  }  }  } |

|  |
| --- |
| Form4.cs |
| public partial class Form4 : Form  {  //마우스 관련 변수  Point fPt;  bool isMove;  private int Form4\_value\_key;//단축키의 순서 값을 저장할 변수  public int Set\_standby\_key  {  get { return this.Form4\_value\_key; } //get 접근자  set { this.Form4\_value\_key = value; } // set 접근자  }  public Form4()  {  InitializeComponent();  //combobox의 디폴트값  comboBox1.Text = "Ctrl+Q";  comboBox1.SelectedText = "Ctrl+Q";  }  //마우스 관련 함수들  private void panel1\_MouseDown(object sender, MouseEventArgs e)  {  isMove = true;  fPt = new Point(e.X, e.Y);  }  private void panel1\_MouseUp(object sender, MouseEventArgs e)  {  isMove = false;  }  private void panel1\_MouseMove(object sender, MouseEventArgs e)  {  if (isMove && (e.Button & MouseButtons.Left) == MouseButtons.Left)  Location = new Point(this.Left - (fPt.X - e.X), this.Top - (fPt.Y - e.Y));  }  //확인 버튼을 누르면  private void button2\_Click(object sender, EventArgs e)  {  //ctrl+q인 경우에는 0값 저장  if (comboBox1.Text == "Ctrl+Q")  {  Set\_standby\_key = 0;  }  else Set\_standby\_key = 1;//ctrl+w인 경우에는 1값 저장  this.Hide();//form 숨김  }  } |

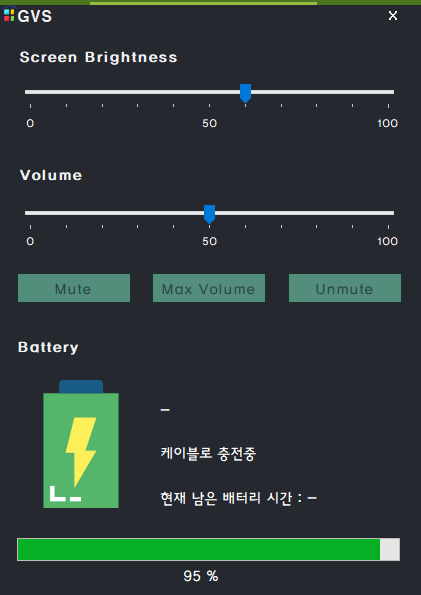
**4.결과/결과분석**



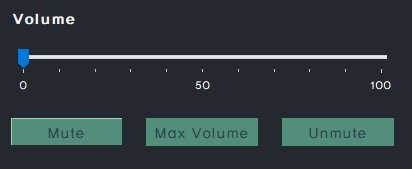
실행 시 메인 화면의 모습이다. 컴퓨터와 파워 세팅 두개의 관리 모드 중 하나를 선택해서 클릭하면 새로운 창이 뜬다.



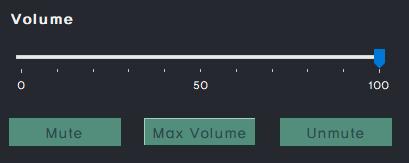
컴퓨터 관리 창의 모습이다. 밝기와 소리를 조절할 수 있는 트랙바와 음소거, 최대음향, 음소거 종료 버튼이 있으며, 배터리에 대한 정보 또한 볼 수 있다.



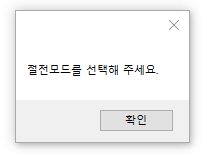
충전 중일 때의 화면이다. 그림과 라벨의 출력 값이 변경된 것을 확인할 수 있다.



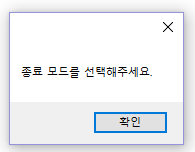
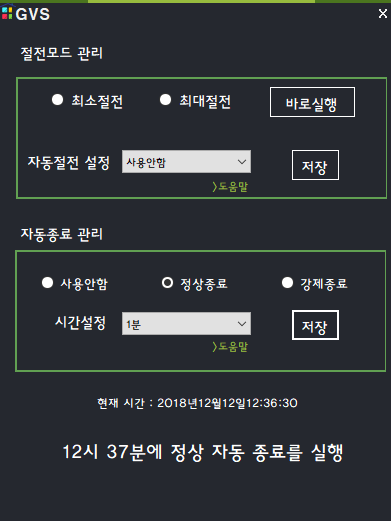
mute버튼을 눌렀을때의 화면이다. 소리의 값이 0으로 간 것을 확인할 수 있다.



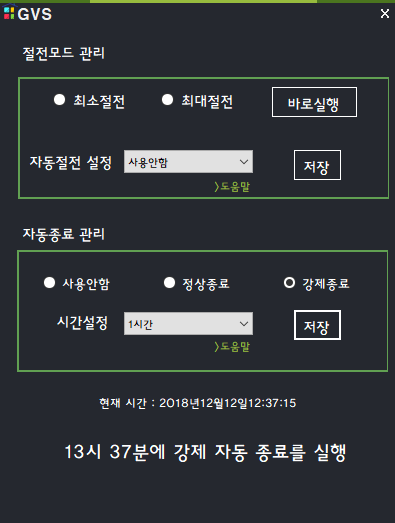
Max volume 버튼을 눌렀을 때의 화면이다. 소리의 값이 100으로 간 것을 확인할 수 있다.



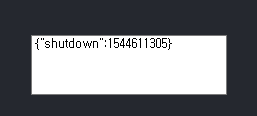
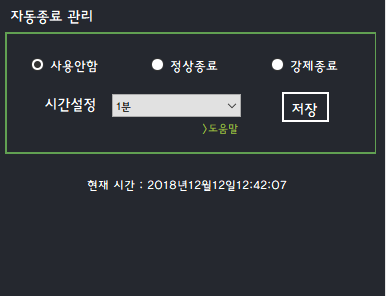
메인 화면에서 Power setting을 눌렀을 때 뜨는 창의 모습이다. 만일, 어떤 절전을 할것인지 선택을 하지않고 바로실행이나 저장 버튼을 누르면 절전모드를 선택해주세요. 라는 메시지박스가 실행된다.



정상종료를 선택하고 시간을 으로 설정하고 저장버튼을 눌렀을 때, 다음과 같이 밑에 언제 정상 자동 종료를 실행할것인지 출력된다. 만일, 어떤 종료를 할것인지 선택하지 않고 저장버튼을 누르면 종료 모드를 선택해주세요. 라는 메시지 박스가 실행된다.



강제종료를 선택한 뒤, 시간을 설정하고 저장을 눌렀을 때, 언제 강제 자동 종료를 실행할지에 대해서 출력된다.

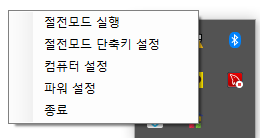


사용안함을 선택하면 다음과 같이 출력되는 것이 사라지고, 자동 종료 또한 취소된다.

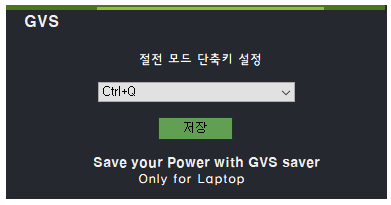
또한 만약, 정상적으로 동작이 되면 textbox에 shutdown을 서버에서 받아와서 출력하게 된다.



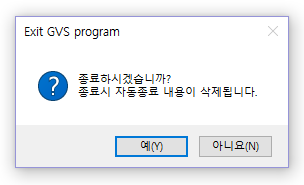
메인 화면을 닫으면 프로그램이 종료가 되지않고, 다음과 같이 윈도우 바에 아이콘이 나타나게 된다. 아이콘을 더블클릭하면 메인 화면이 켜지게 된다.



윈도우바를 오른쪽 마우스를 이용해서 클릭했을 때 다음과 같은 메뉴가 출력된다. 절전모드 실행을 누르면 최소절전모드로 절전모드가 바로 실행된다. 컴퓨터 설정을 누르면 컴퓨터 설정 메뉴창이 나타나고, 파워 설정 버튼을 누르면 파워 설정 메뉴창이 나타나게 된다.



절전모드 단축키 설정버튼을 누르면 다음과 같은 화면이 뜨고, 단축키를 설정한 뒤, 저장을 누르면 그 뒤로부터 단축키를 이용해서 최소절전모드를 실행할 수 있다.



메뉴에서 종료버튼을 누르면 다음과 같은 창이 나타나고, 예를 누르면 프로그램이 종료되면서 자동종료의 내용이 삭제된다.

5.소감

이번 프로젝트에서는 C#을 이용해서 절전관리 프로그램을 구현해보았다. 이번 프로젝트를 진행하면서, 이번 객체윈도우즈 프로그래밍 수업에서 배웠던 여러가지 실습 기능들을 모두 사용할 수 있었다. 지금까지 배운 내용들을 토대로 하나의 프로그램을 만들 수 있어서 좋았다. 특히 powercfg와 같은 dll 파일에 대한 명령어나 기능들을 보다 더 자세하게 공부하게 되었다. 내가 만든 프로그램이긴 하지만, 앞으로도 종종 실행파일로 만들어 친구들에게 나누어 주거나 직접 사용할 수 있다는 생각이 들어서 배운 것도 많고, 좋았던 프로젝트이다.

6.참고

<http://crystalcube.co.kr/45>

<http://usinan.blogspot.com/2018/01/powercfg.html>