ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Морской государственный университет имени адмирала Г.И. Невельского»

Факультет электроники и информационных технологий

Кафедра Вычислительной техники

**Дисциплина «Системы реального времени»**

**Курсовая работа**

«Разработка системы реального времени для мониторинга состояния процессора и видеокарты компьютера»

Выполнил: студент гр. *214.41*

Кисиленко Анастасия Геннадьевна

Проверил: к.т.н., доцент, преподаватель

Маркин. В.Е.

Оглавление

[Введение 3](#_Toc117189481)

[Общее описание проекта 5](#_Toc117189482)

[1. Структура классов 5](#_Toc117189483)

[2. Типы данных и имена переменных 5](#_Toc117189484)

[3. Интерфейс программы 8](#_Toc117189485)

[4. Описания основных процедур 10](#_Toc117189486)

[Алгоритмы 13](#_Toc117189487)

[Код программы 14](#_Toc117189488)

[1. Program.cs 14](#_Toc117189489)

[2. Form1.Designer.cs 14](#_Toc117189490)

[3. Form1.cs 22](#_Toc117189491)

[4. Visitor.cs 30](#_Toc117189492)

[Заключение 31](#_Toc117189493)

[Список литературы 32](#_Toc117189494)

# Введение

В данном документе представлена курсовая работа «Разработка системы реального времени для мониторинга состояния процессора и видеокарты компьютера». Данный проект реализован на языке программирования C# в Visual Studio 2019.

Язык программирования C# в настоящее время набирает популярность среди других языков программирования. И не зря, ведь с помощью C# можно разрабатывать как консольные приложения, приложения с формами и даже веб-приложения. Работать с языком C# можно в Visual Studio, IDE от компании Microsoft, которая как раз занимается разработкой и развитием языка C#. Для разработки проекта я использовала Visual Studio 2019 года. Чтобы получить данные о компонентах компьютера, я использовала библиотеку C# OpenHardwareMonitor. Минусом библиотеки является то, что оно не улавливает AMD видеокарты, а у процессоров этой же марки не улавливает температуру. Однако, большинство видеокарт — это все-таки Nvidia, поэтому свои цели программа может выполнять. А чтобы сделать красивую форму, я использовала библиотеку MetroFramework. В этой библиотеке есть свои компоненты формы и уже заготовленные стили для них.

Система реального времени – система, которая должна реагировать на события в рамках требуемых временных ограничений. Реальное время – количественная характеристика, которая может быть измерена реальными физическими часами.

При разработке проекта были использованы следующие подходы программирования: функциональное, логическое, компонентно-ориентированное, событийно-управляемое и объектно-ориентированное.

В курсовой работе я рассмотрю используемые и реализованные мною функции, методы, классы C#, структура проекта, переменные, интерфейс и алгоритмы.

Целью моей курсовой работы является разработка системы реального времени, которая будет выводить нагрузку и температуру процессора и видеокарты раз в секунду, т.е. в реальном времени.

# Общее описание проекта

## Структура классов

1. Класс **Program**

Точка входа программы. Здесь только активируется окно приложения, а именно стартовая форма, описанная классом Form1;

1. Класс **Form1**

Данный класс является шаблоном для стартового окна системы реального времени. Класс наследуется от MetroFramework.Forms.MetroForm. При запуске программы создается экземпляр данного класса. В этом классе реализован почти весь функционал программы. Примечательно, что для форм MetroFramework есть свои компоненты, но также можно использовать и стандартные компоненты Windows Forms;

1. Класс **Form1.Designer**

Данный класс описывает свойства самой формы и находящиеся на ней компоненты. Компоненты формы представлены объектами. Размер формы фиксированный, 450 на 300 пикселей. Дизайн формы по умолчанию тёмный (Dark), с использованием фиолетового цвета. Пользователь сможет переключить тему программы на светлую в любой момент;

1. Класс **Visitor**

Данный класс нужен для того, чтобы работать с библиотекой OpenHardwareMonitor. Класс наследуется от интерфейса IVisitor. Этот класс используется для того, чтобы реализовать функции «визита» компьютера, его компонентов и т.д.;

## Типы данных и имена переменных

Рассмотрим объекты, которые находятся на форме Form1:

1. MetroButton CloseButton (рис. 1, 1). Кнопка из MetroFramework библиотеки. Данная кнопка реализовывает выход из программы. У кнопки меняется стиль, когда пользователь нажимает на переключатель смены темы.
2. MetroLabel CPULabel (рис. 1, 2). Надпись из MetroFramework, обозначает ориентир для пользователя, что за данные будут справа и снизу. Значение ее параметра Text по умолчанию «Процессор:»
3. MetroLabel DetectedCPULabel (рис. 1, 3). Надпись, которая хранит в себе название определенного процессора в компьютере. Значение параметра Text редактируется при запуске программы, туда выводится название процессора.
4. MetroProgressBar CPUPerfomanceBar (рис. 1, 4). Прогресс-бар, или индикатор прогресса. В моем случае он используется для вывода, насколько нагружен процессор в данный момент времени. Значения целые от 0 до 100%.
5. MetroLabel CPUPercentLabel (рис. 1, 5). Надпись, в ней хранится текущий процент нагрузки процессора. Обновляется надпись каждую секунду.
6. MetroLabel CPUTempLabel (рис. 1, 6). Надпись «Температура:», нужна для обозначения пользователю того, что цифра слева является температурой процессора.
7. Label CurrentTempCPULabel (рис. 1, 7). Обычный Label из Windows Forms. В него записывается температура процессора в данный момент времени.
8. MetroLabel GPULabel (рис. 1, 8). Надпись, которая нужна как индикатор для пользователя. По умолчанию значение text этого объекта = «Видеокарта:». Данная надпись указывает, что слева и снизу будет информация о выбранной видеокарте (их может быть в компьютере несколько).
9. MetroComboBox DetectedGPUBox(рис. 1, 9). Этот объект представляет собой выпадающий список, куда записываются обнаруженные видеокарты в компьютере. Пользователь сможет выбрать видеокарту и посмотреть про неё информацию.
10. MetroProgressBar GPUPerfomanceBar(рис. 1, 10). Индикатор прогресса, используется для обозначения, какая сейчас нагрузка идет на выбранную видеокарту. Значения от 1 до 100.
11. MetroLabel GPUPercentLabel (рис. 1, 11). Надпись, в ней хранится текущий процент нагрузки выбранной видеокарты. Обновляется надпись каждую секунду.
12. MetroLabel GPUTempLabel (рис. 1, 12). Надпись, значение Text которой = «Температура:». Обозначает пользователю, что цифра слева (следующая надпись) является температурой процессора.
13. Label CurrentTempGPULabel (рис. 1, 13). Обычный Label из Windows Forms. В него записывается температура выбранной видеокарты в данный момент времени.
14. MetroLabel DarkThemeLabel (рис. 1, 14). Надпись, обозначающая, что если повернуть переключатель влево, то выберется тёмная тема.
15. MetroLabel LightThemeLabel (рис. 1, 15). Надпись, обозначающая, что если повернуть переключатель вправо, то выберется светлая тема.
16. MetroToggle ThemeToggle (рис. 1, 16). Сам переключатель. В «выключенном» состоянии – тёмная тема, при «включенном» – светлая.

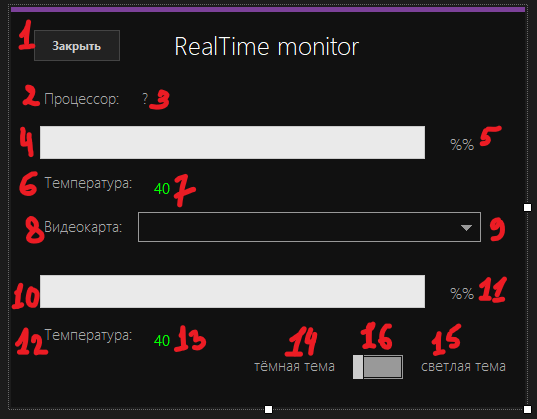


Рисунок 1 Структура Form1

Рассмотрим глобальные переменные в классе Form1:

1. BackgroundWorker Worker. Объект, выполняющий асинхронные операции, не влияющие и не затормаживающие главный поток программы. Используется для того, чтобы выполнять операции каждую секунду асинхронно;
2. Int selectedGPU. В данную переменную записывается текущая выбранная видеокарта. Это нужно для того, чтоб не обращаться к ComboBox из другого потока и использовать Invoke, а просто обратиться к переменной;
3. Bool isShowGPU. Если библиотека не смогла обнаружить какое-либо ГПУ, данная переменная будет являться индикатором, выводить ли блок информации о видеокартах или нет.

## Интерфейс программы

Интерфейс запущенной программы представлен на рисунке 2:

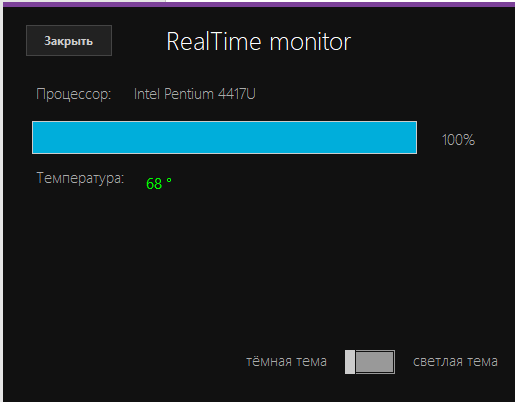


Рисунок Интерфейс программы, обнаружен только процессор

Если пользователь переключит тему, он увидит следующее, рис. 3:

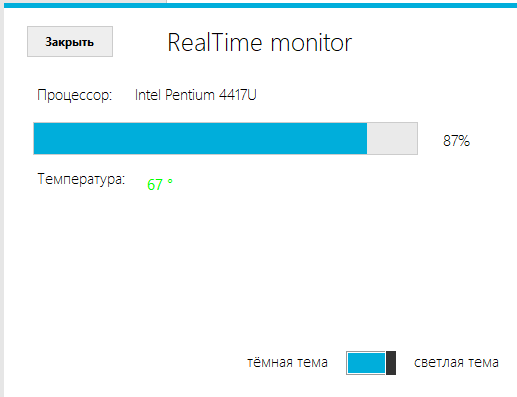


Рисунок Интерфейс программы

А теперь посмотрим, что выдаст программа на компьютере с видеокартой Nvidia, рис. 4:

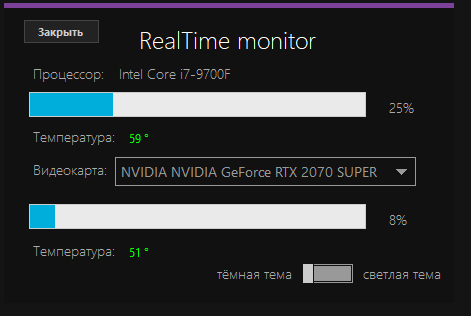


Рисунок Интерфейс программы с другого компьютера

Попробуем запустить какую-то игру и понаблюдаем за нагрузкой и температурой (рис. 5, 6)

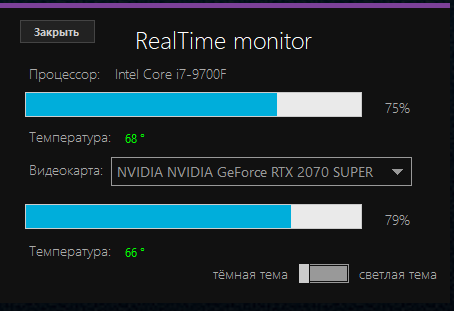


Рисунок Спустя 1 минуту после запуска игры

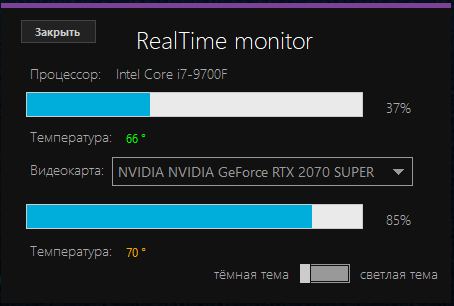


Рисунок Спустя 3 минуты после запуска игры

## Описания основных процедур

Рассмотрим функции и процедуры в классе Form1:

1. Конструктор класса Form1. Без параметров, просто инициализирует все компоненты на форме.
2. private void CloseButton\_Click. Функция, которая вызывается, когда пользователь нажимает на кнопку «Закрыть» (CloseButton).
3. private void ThemeToggle\_CheckedChanged. Функция, вызывающаяся, когда пользователь нажимает на переключатель ThemeToggle (переключает темы).
4. private int GetCPUPercent, входящий параметр Computer computer. В данной функции реализован алгоритм получения нужного компонента компьютера (процессора) и получения нужного сенсора, датчика общей нагрузки процессора.
5. private int GetGPUPercent, входящий параметр Computer computer. В данной функции реализован алгоритм получения нужного компонента компьютера (видеокарты) и получения нужного сенсора, датчика общей нагрузки видеокарты. Примечательно, что в цикле мы ищем ту видеокарту, которая выбрана пользователем в ComboBox, а именно по порядковому номеру.
6. private string GetCPUName, входящий параметр Computer computer. В данной функции мы ищем компонент компьютера CPU и вовращаем его название
7. private List<string> GetGPUNames, входящий параметр Computer computer. В данной функции мы ищем компоненты компьютера GPU и возвращаем список с их названиями.
8. private double GetCPUTemp, входящий параметр Computer computer. Функция ищет нужный компонент компьютера, ищет датчик температуры процессора и выводит значение на датчике.
9. private double GetGPUTemp, входящий параметр Computer computer. Функция ищет нужный компонент компьютера, ищет датчик температуры НУЖНОЙ (выбранной) видеокарты и выводит значение на датчике.
10. private void ViewCurrentCPUTemp. Процедура, которая организованно выводит на форму текущую температуру процессора.
11. private void ViewCurrentGPUTemp. Процедура, которая организованно выводит на форму текущую температуру видеокарты.
12. private void ViewCurrentCPULoad. Процедура, которая организованно выводит на форму текущую нагрузку на процессор.
13. private void ViewCurrentGPULoad. Процедура, которая организованно выводит на форму текущую нагрузку на видеокарте.
14. private void Worker\_DoWork. Функция, реализовывающая асинхронный поток, в котором мы делаем все операции по получению значений с датчиков, а выводим на форму в этом потоке через Invoke.
15. private void Form1\_Load. Когда форма загружается, вызывается данная процедура. Внутри нее мы активируем асинхронный фоновый поток.
16. private void DetectedGPUBox\_SelectedIndexChanged. Процедура, вызывающаяся когда пользователь выбирает из списка другую видеокарту.

# Алгоритмы

Что бы не получали мы, температуру, наименование или нагрузку на CPU/GPU, алгоритм будет в принципе похож. Мы создаем объект класса Computer, включаем ему CPU и GPU, создаем объект класса Visitor (наш класс) и присоединяем к computer. (computer.accept()). После этих манипуляций, мы можем получить Hardware (массив), а выбранный Hardware хранит в себе массив сенсоров (Sensor). Если нам нужно найти определенный компонент, мы ищем его через цикл, сравнивая его тип (HardwareType) с enum. Если нам нужен CPU, то получим HardwareType == HardwareType.CPU. Вывести его имя после получения не составляет труда.

С сенсорами похожая тема, только там мы ищем сенсор температуры, нагрузки и т.д. и выводим его значение.

В зависимости от температуры видеокарты/процессора, выводимый текст окрашивается в зеленый, желтый (больше 70 градусов) и красный (больше 90).

Как уже говорилось выше, получения с датчиков происходят в отдельном потоке асинхронно, а вывод это на компоненты формы происходит через Invoke.

# Код программы

## Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealTimeMonitor

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

## Form1.Designer.cs

namespace RealTimeMonitor

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.CloseButton = new MetroFramework.Controls.MetroButton();

this.CPULabel = new MetroFramework.Controls.MetroLabel();

this.GPULabel = new MetroFramework.Controls.MetroLabel();

this.CPUPerfomanceBar = new MetroFramework.Controls.MetroProgressBar();

this.ThemeToggle = new MetroFramework.Controls.MetroToggle();

this.DarkThemeLabel = new MetroFramework.Controls.MetroLabel();

this.LightThemeLabel = new MetroFramework.Controls.MetroLabel();

this.GPUPerfomanceBar = new MetroFramework.Controls.MetroProgressBar();

this.CPUPercentLabel = new MetroFramework.Controls.MetroLabel();

this.GPUPercentLabel = new MetroFramework.Controls.MetroLabel();

this.DetectedCPULabel = new MetroFramework.Controls.MetroLabel();

this.CPUTempLabel = new MetroFramework.Controls.MetroLabel();

this.GPUTempLabel = new MetroFramework.Controls.MetroLabel();

this.CurrentTempGPULabel = new System.Windows.Forms.Label();

this.CurrentTempCPULabel = new System.Windows.Forms.Label();

this.DetectedGPUBox = new MetroFramework.Controls.MetroComboBox();

this.Worker = new System.ComponentModel.BackgroundWorker();

this.SuspendLayout();

//

// CloseButton

//

this.CloseButton.Font = new System.Drawing.Font("Comic Sans MS", 11.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

this.CloseButton.ForeColor = System.Drawing.Color.White;

this.CloseButton.Location = new System.Drawing.Point(20, 17);

this.CloseButton.Name = "CloseButton";

this.CloseButton.Size = new System.Drawing.Size(75, 23);

this.CloseButton.Style = MetroFramework.MetroColorStyle.Purple;

this.CloseButton.TabIndex = 0;

this.CloseButton.Text = "Закрыть";

this.CloseButton.Theme = MetroFramework.MetroThemeStyle.Dark;

this.CloseButton.Click += new System.EventHandler(this.CloseButton\_Click);

//

// CPULabel

//

this.CPULabel.AutoSize = true;

this.CPULabel.Font = new System.Drawing.Font("Comic Sans MS", 9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.CPULabel.Location = new System.Drawing.Point(25, 61);

this.CPULabel.Name = "CPULabel";

this.CPULabel.Size = new System.Drawing.Size(80, 19);

this.CPULabel.Style = MetroFramework.MetroColorStyle.Purple;

this.CPULabel.TabIndex = 1;

this.CPULabel.Text = "Процессор:";

this.CPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// GPULabel

//

this.GPULabel.AutoSize = true;

this.GPULabel.Font = new System.Drawing.Font("Comic Sans MS", 9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.GPULabel.Location = new System.Drawing.Point(25, 157);

this.GPULabel.Name = "GPULabel";

this.GPULabel.Size = new System.Drawing.Size(83, 19);

this.GPULabel.Style = MetroFramework.MetroColorStyle.Purple;

this.GPULabel.TabIndex = 2;

this.GPULabel.Text = "Видеокарта:";

this.GPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// CPUPerfomanceBar

//

this.CPUPerfomanceBar.Location = new System.Drawing.Point(25, 89);

this.CPUPerfomanceBar.Name = "CPUPerfomanceBar";

this.CPUPerfomanceBar.Size = new System.Drawing.Size(337, 25);

this.CPUPerfomanceBar.Step = 1;

this.CPUPerfomanceBar.TabIndex = 3;

//

// ThemeToggle

//

this.ThemeToggle.AutoSize = true;

this.ThemeToggle.DisplayStatus = false;

this.ThemeToggle.Location = new System.Drawing.Point(299, 261);

this.ThemeToggle.Name = "ThemeToggle";

this.ThemeToggle.Size = new System.Drawing.Size(50, 19);

this.ThemeToggle.Style = MetroFramework.MetroColorStyle.Purple;

this.ThemeToggle.TabIndex = 5;

this.ThemeToggle.Text = "Off";

this.ThemeToggle.Theme = MetroFramework.MetroThemeStyle.Dark;

this.ThemeToggle.UseVisualStyleBackColor = true;

this.ThemeToggle.CheckedChanged += new System.EventHandler(this.ThemeToggle\_CheckedChanged);

//

// DarkThemeLabel

//

this.DarkThemeLabel.AutoSize = true;

this.DarkThemeLabel.Location = new System.Drawing.Point(209, 261);

this.DarkThemeLabel.Name = "DarkThemeLabel";

this.DarkThemeLabel.Size = new System.Drawing.Size(84, 19);

this.DarkThemeLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.DarkThemeLabel.TabIndex = 6;

this.DarkThemeLabel.Text = "тёмная тема";

this.DarkThemeLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// LightThemeLabel

//

this.LightThemeLabel.AutoSize = true;

this.LightThemeLabel.Location = new System.Drawing.Point(355, 261);

this.LightThemeLabel.Name = "LightThemeLabel";

this.LightThemeLabel.Size = new System.Drawing.Size(87, 19);

this.LightThemeLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.LightThemeLabel.TabIndex = 7;

this.LightThemeLabel.Text = "светлая тема";

this.LightThemeLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// GPUPerfomanceBar

//

this.GPUPerfomanceBar.Location = new System.Drawing.Point(25, 201);

this.GPUPerfomanceBar.Name = "GPUPerfomanceBar";

this.GPUPerfomanceBar.Size = new System.Drawing.Size(337, 25);

this.GPUPerfomanceBar.TabIndex = 8;

//

// CPUPercentLabel

//

this.CPUPercentLabel.AutoSize = true;

this.CPUPercentLabel.Location = new System.Drawing.Point(381, 95);

this.CPUPercentLabel.Name = "CPUPercentLabel";

this.CPUPercentLabel.Size = new System.Drawing.Size(31, 19);

this.CPUPercentLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.CPUPercentLabel.TabIndex = 9;

this.CPUPercentLabel.Text = "%%";

this.CPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// GPUPercentLabel

//

this.GPUPercentLabel.AutoSize = true;

this.GPUPercentLabel.Location = new System.Drawing.Point(381, 207);

this.GPUPercentLabel.Name = "GPUPercentLabel";

this.GPUPercentLabel.Size = new System.Drawing.Size(31, 19);

this.GPUPercentLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.GPUPercentLabel.TabIndex = 10;

this.GPUPercentLabel.Text = "%%";

this.GPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// DetectedCPULabel

//

this.DetectedCPULabel.AutoSize = true;

this.DetectedCPULabel.Location = new System.Drawing.Point(111, 61);

this.DetectedCPULabel.Name = "DetectedCPULabel";

this.DetectedCPULabel.Size = new System.Drawing.Size(15, 19);

this.DetectedCPULabel.Style = MetroFramework.MetroColorStyle.Purple;

this.DetectedCPULabel.TabIndex = 11;

this.DetectedCPULabel.Text = "?";

this.DetectedCPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// CPUTempLabel

//

this.CPUTempLabel.AutoSize = true;

this.CPUTempLabel.Location = new System.Drawing.Point(25, 124);

this.CPUTempLabel.Name = "CPUTempLabel";

this.CPUTempLabel.Size = new System.Drawing.Size(91, 19);

this.CPUTempLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.CPUTempLabel.TabIndex = 13;

this.CPUTempLabel.Text = "Температура:";

this.CPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// GPUTempLabel

//

this.GPUTempLabel.AutoSize = true;

this.GPUTempLabel.Location = new System.Drawing.Point(25, 238);

this.GPUTempLabel.Name = "GPUTempLabel";

this.GPUTempLabel.Size = new System.Drawing.Size(91, 19);

this.GPUTempLabel.Style = MetroFramework.MetroColorStyle.Purple;

this.GPUTempLabel.TabIndex = 14;

this.GPUTempLabel.Text = "Температура:";

this.GPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

//

// CurrentTempGPULabel

//

this.CurrentTempGPULabel.AutoSize = true;

this.CurrentTempGPULabel.ForeColor = System.Drawing.Color.Lime;

this.CurrentTempGPULabel.Location = new System.Drawing.Point(122, 242);

this.CurrentTempGPULabel.Name = "CurrentTempGPULabel";

this.CurrentTempGPULabel.Size = new System.Drawing.Size(19, 15);

this.CurrentTempGPULabel.TabIndex = 15;

this.CurrentTempGPULabel.Text = "40";

//

// CurrentTempCPULabel

//

this.CurrentTempCPULabel.AutoSize = true;

this.CurrentTempCPULabel.ForeColor = System.Drawing.Color.Lime;

this.CurrentTempCPULabel.Location = new System.Drawing.Point(122, 128);

this.CurrentTempCPULabel.Name = "CurrentTempCPULabel";

this.CurrentTempCPULabel.Size = new System.Drawing.Size(19, 15);

this.CurrentTempCPULabel.TabIndex = 16;

this.CurrentTempCPULabel.Text = "40";

//

// DetectedGPUBox

//

this.DetectedGPUBox.FormattingEnabled = true;

this.DetectedGPUBox.ItemHeight = 23;

this.DetectedGPUBox.Location = new System.Drawing.Point(111, 154);

this.DetectedGPUBox.Name = "DetectedGPUBox";

this.DetectedGPUBox.Size = new System.Drawing.Size(301, 29);

this.DetectedGPUBox.Style = MetroFramework.MetroColorStyle.Purple;

this.DetectedGPUBox.TabIndex = 17;

this.DetectedGPUBox.Theme = MetroFramework.MetroThemeStyle.Dark;

this.DetectedGPUBox.SelectedIndexChanged += new System.EventHandler(this.DetectedGPUBox\_SelectedIndexChanged);

//

// Worker

//

this.Worker.DoWork += new System.ComponentModel.DoWorkEventHandler(this.Worker\_DoWork);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(450, 300);

this.ControlBox = false;

this.Controls.Add(this.DetectedGPUBox);

this.Controls.Add(this.CurrentTempCPULabel);

this.Controls.Add(this.CurrentTempGPULabel);

this.Controls.Add(this.GPUTempLabel);

this.Controls.Add(this.CPUTempLabel);

this.Controls.Add(this.DetectedCPULabel);

this.Controls.Add(this.GPUPercentLabel);

this.Controls.Add(this.CPUPercentLabel);

this.Controls.Add(this.GPUPerfomanceBar);

this.Controls.Add(this.LightThemeLabel);

this.Controls.Add(this.DarkThemeLabel);

this.Controls.Add(this.ThemeToggle);

this.Controls.Add(this.CPUPerfomanceBar);

this.Controls.Add(this.GPULabel);

this.Controls.Add(this.CPULabel);

this.Controls.Add(this.CloseButton);

this.MaximumSize = new System.Drawing.Size(450, 300);

this.MinimumSize = new System.Drawing.Size(450, 300);

this.Name = "Form1";

this.Resizable = false;

this.ShadowType = MetroFramework.Forms.MetroForm.MetroFormShadowType.None;

this.ShowIcon = false;

this.StartPosition = System.Windows.Forms.FormStartPosition.WindowsDefaultLocation;

this.Style = MetroFramework.MetroColorStyle.Purple;

this.Text = "RealTime monitor";

this.TextAlign = System.Windows.Forms.VisualStyles.HorizontalAlign.Center;

this.Theme = MetroFramework.MetroThemeStyle.Dark;

this.TopMost = true;

this.Load += new System.EventHandler(this.Form1\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private MetroFramework.Controls.MetroButton CloseButton;

private MetroFramework.Controls.MetroLabel CPULabel;

private MetroFramework.Controls.MetroLabel GPULabel;

private MetroFramework.Controls.MetroProgressBar CPUPerfomanceBar;

private MetroFramework.Controls.MetroToggle ThemeToggle;

private MetroFramework.Controls.MetroLabel DarkThemeLabel;

private MetroFramework.Controls.MetroLabel LightThemeLabel;

private MetroFramework.Controls.MetroProgressBar GPUPerfomanceBar;

private MetroFramework.Controls.MetroLabel CPUPercentLabel;

private MetroFramework.Controls.MetroLabel GPUPercentLabel;

private MetroFramework.Controls.MetroLabel DetectedCPULabel;

private MetroFramework.Controls.MetroLabel CPUTempLabel;

private MetroFramework.Controls.MetroLabel GPUTempLabel;

private System.Windows.Forms.Label CurrentTempGPULabel;

private System.Windows.Forms.Label CurrentTempCPULabel;

private MetroFramework.Controls.MetroComboBox DetectedGPUBox;

private System.ComponentModel.BackgroundWorker Worker;

}

}

## Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using OpenHardwareMonitor.Hardware;

using System.Threading;

using System.Windows.Forms;

namespace RealTimeMonitor

{

public partial class Form1 : MetroFramework.Forms.MetroForm

{

private int selectedGPU = 0;

private bool isShowGPU = true;

public Form1()

{

InitializeComponent();

DetectedGPUBox.Items.Clear();

Visitor visitor = new Visitor();

Computer computer = new Computer();

computer.Open();

computer.CPUEnabled = true;

computer.GPUEnabled = true;

computer.Accept(visitor);

DetectedCPULabel.Text = GetCPUName(computer);

foreach (string gpu in GetGPUNames(computer))

{

DetectedGPUBox.Items.Add(gpu);

}

if (DetectedGPUBox.Items.Count > 0)

{

DetectedGPUBox.SelectedIndex = 0;

} else

{

GPULabel.Visible = false;

GPUPercentLabel.Visible = false;

GPUTempLabel.Visible = false;

CurrentTempGPULabel.Visible = false;

DetectedGPUBox.Visible = false;

GPUPerfomanceBar.Visible = false;

isShowGPU = false;

}

ViewCurrentCPULoad(GetCPUPercent(computer));

if (isShowGPU)

{

ViewCurrentGPULoad(GetGPUPercent(computer));

ViewCurrentGPUTemp(GetGPUTemp(computer));

}

ViewCurrentCPUTemp(GetCPUTemp(computer));

computer.Close();

}

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

{

this.Close();

}

private void ThemeToggle\_CheckedChanged(object sender, EventArgs e)

{

if (ThemeToggle.Checked)

{

this.Theme = MetroFramework.MetroThemeStyle.Light;

this.Style = MetroFramework.MetroColorStyle.Blue;

ThemeToggle.Theme = MetroFramework.MetroThemeStyle.Light;

ThemeToggle.Style = MetroFramework.MetroColorStyle.Blue;

CloseButton.Theme = MetroFramework.MetroThemeStyle.Light;

CloseButton.Style = MetroFramework.MetroColorStyle.Blue;

DarkThemeLabel.Theme = MetroFramework.MetroThemeStyle.Light;

DarkThemeLabel.Style = MetroFramework.MetroColorStyle.Blue;

LightThemeLabel.Theme = MetroFramework.MetroThemeStyle.Light;

LightThemeLabel.Style = MetroFramework.MetroColorStyle.Blue;

DetectedCPULabel.Theme = MetroFramework.MetroThemeStyle.Light;

DetectedCPULabel.Style = MetroFramework.MetroColorStyle.Blue;

DetectedGPUBox.Theme = MetroFramework.MetroThemeStyle.Light;

DetectedGPUBox.Style = MetroFramework.MetroColorStyle.Blue;

CPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Light;

CPUPercentLabel.Style = MetroFramework.MetroColorStyle.Blue;

GPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Light;

GPUPercentLabel.Style = MetroFramework.MetroColorStyle.Blue;

CPULabel.Theme = MetroFramework.MetroThemeStyle.Light;

CPULabel.Style = MetroFramework.MetroColorStyle.Blue;

GPULabel.Theme = MetroFramework.MetroThemeStyle.Light;

GPULabel.Style = MetroFramework.MetroColorStyle.Blue;

CPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Light;

CPUTempLabel.Style = MetroFramework.MetroColorStyle.Blue;

GPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Light;

GPUTempLabel.Style = MetroFramework.MetroColorStyle.Blue;

CurrentTempCPULabel.BackColor = this.BackColor;

CurrentTempGPULabel.BackColor = this.BackColor;

}

else

{

this.Theme = MetroFramework.MetroThemeStyle.Dark;

this.Style = MetroFramework.MetroColorStyle.Purple;

ThemeToggle.Theme = MetroFramework.MetroThemeStyle.Dark;

ThemeToggle.Style = MetroFramework.MetroColorStyle.Purple;

CloseButton.Theme = MetroFramework.MetroThemeStyle.Dark;

CloseButton.Style = MetroFramework.MetroColorStyle.Purple;

DarkThemeLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

DarkThemeLabel.Style = MetroFramework.MetroColorStyle.Purple;

LightThemeLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

LightThemeLabel.Style = MetroFramework.MetroColorStyle.Purple;

DetectedCPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

DetectedCPULabel.Style = MetroFramework.MetroColorStyle.Purple;

DetectedGPUBox.Theme = MetroFramework.MetroThemeStyle.Dark;

DetectedGPUBox.Style = MetroFramework.MetroColorStyle.Purple;

CPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

CPUPercentLabel.Style = MetroFramework.MetroColorStyle.Purple;

GPUPercentLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

GPUPercentLabel.Style = MetroFramework.MetroColorStyle.Purple;

CPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

CPULabel.Style = MetroFramework.MetroColorStyle.Purple;

GPULabel.Theme = MetroFramework.MetroThemeStyle.Dark;

GPULabel.Style = MetroFramework.MetroColorStyle.Purple;

CPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

CPUTempLabel.Style = MetroFramework.MetroColorStyle.Purple;

GPUTempLabel.Theme = MetroFramework.MetroThemeStyle.Dark;

GPUTempLabel.Style = MetroFramework.MetroColorStyle.Purple;

CurrentTempCPULabel.BackColor = this.BackColor;

CurrentTempGPULabel.BackColor = this.BackColor;

}

}

private int GetCPUPercent(Computer computer)

{

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.CPU)

{

for (int j = 0; j < computer.Hardware[i].Sensors.Length; j++)

{

if (computer.Hardware[i].Sensors[j].SensorType == SensorType.Load)

{

if (computer.Hardware[i].Sensors[j].Name == "CPU Total")

{

return (int)computer.Hardware[i].Sensors[j].Value;

}

}

}

}

}

return 0;

}

private int GetGPUPercent(Computer computer)

{

int c = 0;

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.GpuNvidia || computer.Hardware[i].HardwareType == HardwareType.GpuAti)

{

if (c == selectedGPU)

{

for (int j = 0; j < computer.Hardware[i].Sensors.Length; j++)

{

if (computer.Hardware[i].Sensors[j].SensorType == SensorType.Load)

{

if (computer.Hardware[i].Sensors[j].Name == "GPU Core")

{

return (int)computer.Hardware[i].Sensors[j].Value;

}

}

}

}

c++;

}

}

return 0;

}

private string GetCPUName(Computer computer)

{

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.CPU)

{

return computer.Hardware[i].Name;

}

}

return "";

}

private List<string> GetGPUNames(Computer computer)

{

List<string> gpus = new List<string>();

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.GpuNvidia)

{

gpus.Add(computer.Hardware[i].Name);

}

else if (computer.Hardware[i].HardwareType == HardwareType.GpuAti)

{

gpus.Add(computer.Hardware[i].Name);

}

}

return gpus;

}

private double GetCPUTemp(Computer computer)

{

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.CPU)

{

for (int j = 0; j < computer.Hardware[i].Sensors.Length; j++)

{

if (computer.Hardware[i].Sensors[j].SensorType == SensorType.Temperature)

{

if (computer.Hardware[i].Sensors[j].Name == "CPU Package")

{

return (double)computer.Hardware[i].Sensors[j].Value;

}

}

}

}

}

return 0;

}

private double GetGPUTemp(Computer computer)

{

int c = 0;

for (int i = 0; i < computer.Hardware.Length; i++)

{

if (computer.Hardware[i].HardwareType == HardwareType.GpuNvidia || computer.Hardware[i].HardwareType == HardwareType.GpuAti)

{

if (c == selectedGPU)

{

for (int j = 0; j < computer.Hardware[i].Sensors.Length; j++)

{

if (computer.Hardware[i].Sensors[j].SensorType == SensorType.Temperature)

{

if (computer.Hardware[i].Sensors[j].Name == "GPU Core")

{

return (double)computer.Hardware[i].Sensors[j].Value;

}

}

}

}

c++;

}

}

return 0;

}

private void ViewCurrentCPUTemp(double temp)

{

if (temp < 70)

{

CurrentTempCPULabel.ForeColor = Color.Lime;

}

else if (temp < 90)

{

CurrentTempCPULabel.ForeColor = Color.Orange;

}

else

{

CurrentTempCPULabel.ForeColor = Color.Red;

}

CurrentTempCPULabel.Text = temp.ToString() + " °";

}

private void ViewCurrentGPUTemp(double temp)

{

if (temp < 70)

{

CurrentTempGPULabel.ForeColor = Color.Lime;

}

else if (temp < 90)

{

CurrentTempGPULabel.ForeColor = Color.Orange;

}

else

{

CurrentTempGPULabel.ForeColor = Color.Red;

}

CurrentTempGPULabel.Text = temp.ToString() + " °";

}

private void ViewCurrentCPULoad(int cpu)

{

CPUPerfomanceBar.Value = cpu;

CPUPercentLabel.Text = cpu + "%";

}

private void ViewCurrentGPULoad(int gpu)

{

GPUPerfomanceBar.Value = gpu;

GPUPercentLabel.Text = gpu + "%";

}

private void Worker\_DoWork(object sender, DoWorkEventArgs e)

{

while (true)

{

Visitor visitor = new Visitor();

Computer computer = new Computer();

computer.Open();

computer.CPUEnabled = true;

computer.GPUEnabled = true;

computer.Accept(visitor);

int cpu = GetCPUPercent(computer);

int gpu = 0;

double gpuTemp = 0;

if (isShowGPU)

{

gpu = GetGPUPercent(computer);

gpuTemp = GetGPUTemp(computer);

}

double cpuTemp = GetCPUTemp(computer);

computer.Close();

Action act = () =>

{

if (isShowGPU)

{

ViewCurrentGPUTemp(gpuTemp);

ViewCurrentGPULoad(gpu);

}

ViewCurrentCPUTemp(cpuTemp);

ViewCurrentCPULoad(cpu);

};

this.Invoke(act);

Thread.Sleep(1000);

}

}

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

{

Worker.RunWorkerAsync();

}

private void DetectedGPUBox\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedGPU = DetectedGPUBox.SelectedIndex;

}

}

}

## Visitor.cs

using System;

using System.Collections.Generic;

using System.Text;

using OpenHardwareMonitor.Hardware;

namespace RealTimeMonitor

{

class Visitor : IVisitor

{

public void VisitComputer(IComputer computer)

{

computer.Traverse(this);

}

public void VisitHardware(IHardware hardware)

{

hardware.Update();

foreach(IHardware hw in hardware.SubHardware)

{

hw.Accept(this);

}

}

public void VisitParameter(IParameter parameter)

{

}

public void VisitSensor(ISensor sensor)

{

}

}

}

# Заключение

В ходе курсовой работы я разработала систему реального времени, мониторинг статуса видеокарт и процессора компьютера на языке программирования C#. В программе пользователь может в реальном времени следить за температурой и нагрузкой своих компонентов. Для пользователя реализована возможность сменить тему приложения, светлая и темная. Программа может быть очень полезна для любого из пользователей, если им нужно отслеживать статус компонентов своего компьютера в реальном времени.

# Список литературы

1. Краткий обзор языка C# [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (дата обращения: 19.10.2022).
2. Microsoft Visual Studio [Электронный ресурс]. — URL: <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio> (дата обращения: 19.10.2022).
3. Система реального времени [Электронный ресурс]. — URL: <https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8> (дата обращения: 19.10.2022).
4. OpenHardwareMonitor класс [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/iot.device.hardwaremonitor.openhardwaremonitor?view=iot-dotnet-1.3> (дата обращения: 19.10.2022).
5. Парадигма программирования [Электронный ресурс]. — URL: <https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F> (дата обращения: 19.10.2022).
6. Парадигма программирования [Электронный ресурс]. — URL: <https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F> (дата обращения: 19.10.2022).