Пермский филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский университет   
«Высшая школа экономики»

**Факультет профессиональной переподготовки**

**Разработка информационной системы контроля мгновенного расхода топлива автомобиля**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Проектная работа по направлению подготовки

«Объектно-ориентированное программирование»

Слушателя Шабурова Ильи Алексеевича

группы ООП-19

Руководитель проектной работы Старший преподаватель

кафедры информационных

технологий в бизнесе

Ланин В. В.

Пермь, 2020

Аннотация

Автор – Шабуров Илья Алексеевич.

Тема работы – «Разработка информационной системы контроля мгновенного расхода топлива автомобиля».

Год издания – 2020.

Издательство – Факультет профессиональной переподготовки НИУ ВШЭ – Пермь.

Количество глав – 3.

В итоговой работе рассмотрены вопросы, связанные с работой и устройством информационной системы контроля мгновенного расхода топлива автомобиля, а также требования, на основе которых выполнена реализация программного продукта.

Работа содержит \_\_ страниц основного текста, состоит из 3 глав и \_\_ приложений.

Оглавление

[**Введение** 4](#_Toc40008772)

[Глава 1. Анализ задачи и разработка требований к системе «Климат-контроль» 5](#_Toc40008773)

[Глава 2. Проектирование системы 11](#_Toc40008774)

# **Введение**

Автомобили стали неотъемлемой частью жизни человека. С каждым годом их количество на дорогах мира растет, а вместе с ними и растет количество вредных выбросов в атмосферу. Ученые и инженеры работают над повышением экологичности вновь выпускаемых машин, но что же можно сделать с уже выпущенными автомобилями?

Информационная система контроля мгновенного расхода топлива автомобиля позволяет получить информацию о расходе топлива автомобиля в настоящий момент времени, и на основе полученной информации корректировать работу двигателя в сторону более экономичной езды, а с этим и более экологичной.

Целью итоговой проектной работы является разработка информационной системы контроля мгновенного расхода топлива автомобиля для пользователей автомобилей, оснащенных двигателем внутреннего сгорания с микропроцессорным управлением (все современные двигатели) и способных передавать данные в диагностический разъем.

Для достижения указанной цели в работе поставлены следующие задачи:

- анализ разрабатываемой системы;

- формирование требований к разрабатываемой системе;

- проектирование системы;

- разработка системы;

- практический опыт разработки приложений и закрепление знаний, полученных в рамках всей учебной программы.

Для решения поставленных задач были использованы следующие инструменты: язык программирования C#, унифицированный язык моделирования UML, диагностический OBD-II Bluetooth сканер с чипом ELM327 v1.5, а также Bluetooth 4.0 адаптер для ПК.

Глава 1. Анализ задачи и разработка требований к информационной системе контроля мгновенного расхода топлива

Целевой аудиторией информационной системы контроля мгновенного расхода топлива – далее информационная система, являются пользователи автомобилей с двигателем внутреннего сгорания. Указанный двигатель должен быть оснащен микропроцессорным управлением, а также различными датчиками, информация с которых необходима электронному блоку управления двигателем (ЭБУ) для организации правильного режима работы двигателя.

Диагностический Bluetooth сканер имеющий чип ELM327 v1.5 далее – диагностический сканер, является распространенным устройством бюджетного сегмента диагностических сканеров, совмещающий в себе отличную функциональность за приемлемую цену. Сканер подключается в диагностический разъем автомобиля и способен получить данные с ЭБУ автомобиля, в том числе информацию с различных датчиков двигателя по стандарту OBD-II.

Стандарт OBD-II был разработан в середине 90х годов и позволяет проводить мониторинг систем управления двигателя.

## Формирование требований к разрабатываемой системе

1. Информационная система должна взаимодействовать с диагностическим сканером через последовательный порт и получать данные с ЭБУ автомобиля.
2. Информационная система должна иметь на главном экране кнопки «подключиться» к диагностическому сканеру и «отключиться» от диагностического устройства.
3. Информационная система должна отображать на основном экране информацию о мгновенном расходе топлива, в случае если автомобиль стоит на месте в л/час.
4. Информационная система должна отображать на основном экране информацию о мгновенном расходе топлива, в случае если автомобиль движется в л/100км
5. Частота обновления данных должна составлять минимум 1 раз в секунду.
6. На главном экране должен отображаться график: ось абсцисс – 10 промежутков по 6 секунд, в общей сложности 1 минута, ось ординат – усредненный мгновенный расхода топлива за 6 секунд.
7. Если ЭБУ не предоставляет рассчитанную им информацию о мгновенном расходе топлива, информационная система должна рассчитать, основываясь на данных с датчиков массового расхода воздуха или времени открытия форсунок и расчетной производительности.
8. Информационная система должна содержать меню настроек, в котором выбирается устройство, к которому подключается информационная система.
9. Меню настроек должно иметь поле ввода пользователем расчетной производительности форсунок.

## Формализация требований

Изображение выглядит как текст, карта

Автоматически созданное описаниеНа этапе формализации требований к системе обратимся к диаграмме прецедентов, представленной на рисунке 1.1:

*Рис.1.1. Диаграмма прецедентов*

У информационной системы есть только один актор – пользователь.

Далее приведено более подробное описание потоков:

1. Название: настройка информационной системы, выбор устройства для подключения

Актор: Пользователь

Краткое описание процесса: Пользователь выбирает диагностический сканер из списка, к которому должна подключиться информационная система

Триггер: открытие настроек информационной системы

Осн. Поток:

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь нажимает кнопку открытия настроек информационной системы | 1. Система отображает настройки |
| 1. Пользователь нажимает на выпадающий список доступных диагностических сканеров | 1. Система производит поиск устройств. |
|  | 1. Система добавляет устройства в список |
|  | 1. Система отображает список доступных устройств |
| 1. Пользователь выбирает устройство из списка | 1. Система сохраняет устройство в памяти, а также в рабочем файле. |
| 1. Пользователь возвращается на основной экран | 1. Система отрисовывает основной экран с активной кнопкой «Подключить» |

Альтернативный Поток:

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь нажимает кнопку открытия настроек информационной системы | 1. Система отображает настройки |
| 1. Пользователь нажимает на выпадающий список доступных диагностических сканеров | 1. Система производит поиск окружающих устройств. |
|  | 1. Окружающие устройства не обнаружены |
|  | 1. Система выводит сообщение об отсутствии окружающих устройств |
| 1. Пользователь закрывает настройки | 1. Система отрисовывает основной экран с неактивной кнопкой «Подключить» |

1. Название: настройка информационной системы, установка производительности форсунок

Актор: Пользователь

Краткое описание процесса: Пользователь вводит производительность форсунок

Триггер: открытие настроек информационной системы

Осн. Поток:

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь нажимает кнопку открытия настроек информационной системы | 1. Система отображает настройки |
| 1. Пользователь вводит производительность форсунок в соответствующее поле | 1. Система сохраняет производительность в памяти, а также в рабочем файле. |
| 1. Пользователь возвращается на основной экран | 1. Система отрисовывает основной экран |

1. Название: подключение к диагностическому сканеру

Актор: Пользователь

Краткое описание процесса: Подключение к диагностическому сканеру, и проверка связи.

Триггер: нажатие на кнопку «Подключить»

Осн. Поток:

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь нажимает кнопку «Подключить» | 1. Система открывает новое соединение с выбранным в настройках устройством |
|  | 1. Система проверяет идентификацию подключенного устройства |
|  | 1. Система устанавливает статус подключения «Подключено» |

1. Название: получение информации о мгновенном расходе топлива

Актор: Пользователь

Краткое описание процесса: после подключения к диагностическому сканеру информационная система начинает с периодичностью «одна секунда» запрашивать у диагностического сканера информацию о мгновенном расходе топлива, или, если эту информацию сканер не предоставляет, информацию о времени открытия форсунок. Также получает информацию о текущей скорости автомобиля, и отображает ее на экране.

Триггер: статус подключения – «Подключено»

Осн. Поток:

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь получает информацию о мгновенном расходе топлива | 1. Система опрашивает диагностический сканер |
|  | 1. Система отображает информацию |
|  | 1. Система строит график |
|  | 1. Система отображает статус движения автомобиля |

## Описание автоматизируемых активностей с помощью диаграммы активностей

![Изображение выглядит как текст

Автоматически созданное описание]()Для более полного понимания бизнес-процессов, происходящих в информационной системе, построим диаграммы активностей. Диаграммы активностей описывают взаимодействие актора и системы, демонстрируя его действия на каждом этапе. Диаграммы представлены на рисунках 1.2 – 1.5.

*Рис.1.2. Диаграмма активностей, запуск информационной системы*

*Рис.1.3. Диаграмма активностей, настройка информационной системы*

*![Изображение выглядит как текст, карта

Автоматически созданное описание]()*

*Рис.1.4. Диаграмма активностей, подключение к диагностическому сканеру* ![Изображение выглядит как текст, карта

Автоматически созданное описание]()

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

*Рис.1.5. Диаграмма активностей, получение информации с диагностического сканера*

Диагностический Bluetooth сканер с чипом ELM327 v1.5 позволяет «общаться» с ним через последовательный порт по Bluetooth соединению, что накладывает некоторые особенности в передаче и получении пакетов. Отправка и получение данных происходит через буферы отправки и приема соответственно. Информация, получаемая от диагностического сканера, может записываться в буфер приема продолжительное время, вплоть до нескольких секунд. В связи с вышесказанным процесс опроса диагностического сканера будет происходить асинхронными сообщениями.

Изображение выглядит как снимок экрана

Автоматически созданное описаниеДля более полного анализа процесса работы информационной системы, рассмотрим прецедент «общения» через нотацию диаграммы последовательностей на рисунке 1.6.

*Рис.1.6. Диаграмма последовательностей, получение информации с диагностического сканера*

Глава 2. Проектирование системы

На рисунке 2.1 представлена диаграмма классов, представляющая собой структуру сущностей системы и отображающая основные понятия, структуру классов и их отношения между собой.

Изображение выглядит как снимок экрана

Автоматически созданное описание

*Рис.2.1. Диаграмма классов*

Рассмотрим каждый класс поподробнее.

2.1. StatusBar

*Табл.2.1. Класс StatusBar*

| Атрибут | Тип атрибута | Назначение |
| --- | --- | --- |
| Id | int | Уникальный идентификатор для поиска сущностей и установки их связей. |
| Email | string | Почта пользователя, используемая в качестве логина |
| Password | string | Пароль для доступа в систему |
| Department | string | Отдел, управление которым доступно пользователю |

2.2. Отдел

Класс Отдел (табл. 2.2.) определяет границы, нарушение которых имеет в виде результата активацию исполнительных устройств.

***Табл.2.2. Класс Отдел***

| Атрибут | Тип атрибута | Назначение |
| --- | --- | --- |
| Id | int | Уникальный идентификатор для поиска сущностей и установки их связей. |
| Name | string | Название отдела |
| HotTemp | int | Верхняя граница температуры, при превышении активируется кондиционер |
| NormTemp | int | Нормальная граница температуры, при пересечении деактивируются кондиционер или обогреватель |
| ColdTemp | int | Нижняя граница температуры, при снижении ниже активируется обогреватель |
| CO2High | int | Верхняя граница загазованности, при превышении активируется вентиляция |
| CO2Norm | int | Нормальная граница загазованности, при снижении ниже деактивируется вентиляция |

2.3. Датчик

Класс Датчик (табл. 2.3.) содержит в себе все возможные типы и адреса датчиков, задействованных в системе управления климатом, а также последние сохраненные значения датчиков.

***Табл.2.3. Класс Датчик***

| Атрибут | Тип атрибута | Назначение |
| --- | --- | --- |
| Id | int | Уникальный идентификатор для поиска сущностей и установки их связей. |
| Name | string | Название датчика |
| Type | string | Тип датчика (температура или CO2) |
| DepartmentId | string | Принадлежность отделу |
| Value | int | Показания датчика |

2.4. Устройство

Класс Устройство (табл. 2.4.) содержит в себе все возможные типы и адреса исполнительных устройств, задействованных в системе управления климатом, а также их текущее состояние (вкл./выкл.).

***Табл.2.4. Класс Устройство***

| Атрибут | Тип атрибута | Назначение |
| --- | --- | --- |
| Id | int | Уникальный идентификатор для поиска сущностей и установки их связей. |
| Name | string | Название датчика |
| Type | string | Тип устройства (кондиционер, вентиляция, обогреватель) |
| DepartmentId | string | Принадлежность отделу |
| TriggerState | int | Текущее состояние устройства |

**Глава 3. Реализация системы управления микроклиматом**

| № теста | Начальное состояние | Внешние действия | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Обогреватель выключен | Показания датчика температуры между нормальным уровнем и нижним | Отсутствие сработки обогрева |
| 2 | Обогреватель выключен | Показания датчика температуры меньше нижнего уровня | Включение обогревателя |
| 3 | Обогреватель включен | Показания датчика температуры между нормальным уровнем и нижним | Сохранение включенного обогрева |
| 4 | Обогреватель включен | Показания датчика температуры выше нормального уровня | Выключение обогревателя |
| 5 | Кондиционер выключен | Показания датчика температуры между нормальным уровнем и верхним | Отсутствие сработки кондиционера |
| 6 | Кондиционер выключен | Показания датчика температуры больше верхнего уровня | Включение кондиционера |
| 7 | Кондиционер включен | Показания датчика температуры между нормальным уровнем и верхним | Сохранение включенного кондиционера |
| 8 | Кондиционер включен | Показания датчика температуры ниже нормального уровня | Выключение кондиционера |
| 9 | Вентиляция выключена | Показания датчика загазованности между нормальным уровнем и верхним | Отсутствие сработки вентиляции |
| 10 | Вентиляция выключена | Показания датчика загазованности больше верхнего уровня | Включение вентиляции |
| 11 | Вентиляция включена | Показания датчика загазованности между нормальным уровнем и верхним | Сохранение включенной вентиляции |
| 12 | Вентиляция включена | Показания датчика загазованности ниже нормального уровня | Выключение вентиляции |

**Приложениe А. Листинг 1. Класс HomeController**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using FinalProject.Models;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.AspNet.Identity;

using FinalProject.Models;

namespace FinalProject.Controllers

{

public class HomeController : Controller

{

ClimateContext db = new ClimateContext();

public ActionResult Index()

{

var departments = db.Departments;

ViewBag.Departments = departments;

return View();

}

[HttpGet]

public ActionResult Edit(int id)

{

ViewBag.DepartmentId = id;

return View();

}

[HttpPost]

public string Edit(EditDep editDep)

{

db.Edits.Add(editDep);

db.SaveChanges();

return "Ваши изменения сохранены";

}

public ActionResult GetUsers()

{

List<ApplicationUser> users = new List<ApplicationUser>();

using (ApplicationDbContext db = new ApplicationDbContext())

{

users = db.Users.ToList();

}

return View(users);

}

[Authorize]

public ActionResult GetMyRoles()

{

IList<string> roles = new List<string> { "Роль не определена" };

ApplicationUserManager userManager = HttpContext.GetOwinContext()

.GetUserManager<ApplicationUserManager>();

ApplicationUser user = userManager.FindByEmail(User.Identity.Name);

if (user != null)

roles = userManager.GetRoles(user.Id);

return View(roles);

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

}**Приложениe Б. Листинг 2. Модель IdentityModels**

using System.Data.Entity;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

namespace FinalProject.Models

{

// You can add profile data for the user by adding more properties to your ApplicationUser class, please visit https://go.microsoft.com/fwlink/?LinkID=317594 to learn more.

public class ApplicationUser : IdentityUser

{

public string Department { get; set; }

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Обратите внимание, что authenticationType должен совпадать с типом, определенным в CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Здесь добавьте утверждения пользователя

return userIdentity;

}

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection", throwIfV1Schema: false)

{

}

public DbSet<Department> Departments { get; set; }

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

}

public class AppDbInitializer : DropCreateDatabaseAlways<ApplicationDbContext>

{

protected override void Seed(ApplicationDbContext context)

{

var UserManager = new ApplicationUserManager(new UserStore<ApplicationUser>(context));

var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

var role1 = new IdentityRole { Name = "admin" };

var role2 = new IdentityRole { Name = "user" };

RoleManager.Create(role1);

RoleManager.Create(role2);

var admin = new ApplicationUser { Email = "admin@admin.ru", UserName = "admin@admin.ru", Department = "Техподдержка"};

string password = "Qwe123!";

var result = UserManager.Create(admin, password);

if (result.Succeeded)

{

UserManager.AddToRole(admin.Id, role1.Name);

UserManager.AddToRole(admin.Id, role2.Name);

}

var usertest = new ApplicationUser { Email = "user@user.ru", UserName = "user@user.ru", Department = "Продажи" };

string password1 = "Qwe123!";

var result1 = UserManager.Create(usertest, password1);

if (result1.Succeeded)

{

UserManager.AddToRole(usertest.Id, role2.Name);

}

base.Seed(context);

}

}

}

**Приложениe В. Листинг 3. Класс ClimateContext**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using FinalProject.Models;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.AspNet.Identity;

using FinalProject.Models;

namespace FinalProject.Controllers

{

public class HomeController : Controller

{

ClimateContext db = new ClimateContext();

public ActionResult Index()

{

var departments = db.Departments;

ViewBag.Departments = departments;

return View();

}

[HttpGet]

public ActionResult Edit(int id)

{

ViewBag.DepartmentId = id;

return View();

}

[HttpPost]

public string Edit(EditDep editDep)

{

db.Edits.Add(editDep);

db.SaveChanges();

return "Ваши изменения сохранены";

}

public ActionResult GetUsers()

{

List<ApplicationUser> users = new List<ApplicationUser>();

using (ApplicationDbContext db = new ApplicationDbContext())

{

users = db.Users.ToList();

}

return View(users);

}

[Authorize]

public ActionResult GetMyRoles()

{

IList<string> roles = new List<string> { "Роль не определена" };

ApplicationUserManager userManager = HttpContext.GetOwinContext()

.GetUserManager<ApplicationUserManager>();

ApplicationUser user = userManager.FindByEmail(User.Identity.Name);

if (user != null)

roles = userManager.GetRoles(user.Id);

return View(roles);

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

}

**Приложениe Г. Листинг 4. Приложение обработки данных**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Timers;

namespace ConsoleApp1

{

class Program

{

//проверка на холод

static int TempCheckCold(int heater, int temper, int norm, int low)

{

if (temper < low)

{

heater = 1;

Console.WriteLine("Обогреватель включен");

}

else if (temper > norm)

{

heater = 0;

Console.WriteLine("Обогреватель выключен");

}

return heater;

}

//проверка на жару

static int TempCheckHot(int conditioner, int temper, int high, int norm)

{

if (temper > high)

{

conditioner = 1;

Console.WriteLine("Кондиционер включен");

}

else if (temper < norm)

{

conditioner = 0;

Console.WriteLine("Кондиционер выключен");

}

return conditioner;

}

//проверка на уровень углекислого газа

static int OxygenCheck(int ventilation, int CO2, int high, int norm)

{

if (CO2 > high)

{

ventilation = 1;

Console.WriteLine("Вентиляция включена");

}

else if (CO2 < norm)

{

ventilation = 0;

Console.WriteLine("Вентиляция выключена");

}

return ventilation;

}

static void Main(string[] args)

{

//параметры для тестов

bool trigger = true;

int TempSensorValue = 22;

int CO2SensorValue = 1700;

int HotTemp = 25;

int NormTemp = 23;

int ColdTemp = 21;

int CO2High = 1500;

int CO2Norm = 1000;

int HeaterState = 0;

int ConditionerState = 0;

int VentilationState = 0;

do

{

//передача в приложение текущих параметров из веб - включение системы, текущие показания датчиков температуры и CO2, выставленные границы включения и выключения кондиционера, обогрева и вентиляции для нужного отдела

int temper = TempSensorValue;

int co2 = CO2SensorValue;

int hottemp = HotTemp;

int normtemp = NormTemp;

int coldtemp = ColdTemp;

int co2high = CO2High;

int co2norm = CO2Norm;

int heater = HeaterState;

int conditioner = ConditionerState;

int ventilation = VentilationState;

TempCheckCold(heater, temper, NormTemp, ColdTemp);

TempCheckHot(conditioner, temper, HotTemp, NormTemp);

OxygenCheck(ventilation, co2, CO2High, CO2Norm);

Timer timer = new Timer();

timer.Interval = 60000;

timer.Start();

}

while (trigger == true);

}

}

}

**Приложениe Д. Листинг 5. Класс Department**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace FinalProject.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

public int HotTemp { get; set; }

public int NormTemp { get; set; }

public int ColdTemp { get; set; }

public int CO2High { get; set; }

public int CO2Norm { get; set; }

}

}

**Приложениe E. Листинг 6. Представление Index**

@{

ViewBag.Title = "Страница настройки по отделам";

}

<div>

<h3>Список отделов</h3>

<table>

<tr>

<td><p>Id</p></td>

<td><p>Название отдела</p></td>

<td><p>Верхняя граница температуры</p></td>

<td><p>Нормальное значение температуры</p></td>

<td><p>Нижняя граница температуры</p></td>

<td><p>Верхняя граница уровня CO2</p></td>

<td><p>Нормальное значение уровня CO2</p></td>

<td></td>

</tr>

@foreach(var d in ViewBag.Departments)

{

<tr>

<td><p>@d.Id</p></td>

<td><p>@d.Name</p></td>

<td><p>@d.HotTemp</p></td>

<td><p>@d.NormTemp</p></td>

<td><p>@d.ColdTemp</p></td>

<td><p>@d.CO2High</p></td>

<td><p>@d.CO2Norm</p></td>

<td><p><a href="/Home/Edit@d.Id">Изменить</a></p></td>

</tr>

}

</table>

</div>

**Приложениe Ж. Листинг 7. Представление Edit**

@{

ViewBag.Title = "Редактирование настроек отдела";

}

<div>

<h3>Редактирование настроек отдела</h3>

<form method="post">

<input type="hidden" value="@ViewBag.DepartmentId" name="DepartmentId" />

<table>

<tr>

<td><p>Введите значение верхней границы температуры</p></td>

<td><input type="number" name="HotTemp" /></td>

</tr>

<tr>

<td><p>Введите значение нормальной границы температуры</p></td>

<td><input type="number" name="NormTemp" /></td>

</tr>

<tr>

<td><p>Введите значение нижней границы температуры</p></td>

<td><input type="number" name="ColdTemp" /></td>

</tr>

<tr>

<td><p>Введите значение верхней границы загазованности</p></td>

<td><input type="number" name="CO2High" /></td>

</tr>

<tr>

<td><p>Введите значение нормальной границы загазованности</p></td>

<td><input type="number" name="CO2Norm" /></td>

</tr>

<tr><td><input type="submit" value="Принять изменения" /></td><td></td></tr>

</table>

</form>

</div>