МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ автономное ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



«Программирование алгоритмов систем управления»

Курсовой проект по дисциплине

«Проектирование и разработка программного обеспечения с использованием диаграмм состояний»

Группа 221-328

Студент Шурова Д.С.

Дата 22.12.2023

Преподаватель Идиатуллов Т.Т.

2023

**Задание**

Разработать модель программного обеспечения системы управления мобильным сервисным роботом на основе метода построения диаграмм состояний.

С использованием симулятора мобильного робота IoTRobotWorld разработать систему управления роботом (на языке C#) на основе подготовленной модели.

Перед выполнением задания необходимо подготовить файл застройки рабочей зоны (поля) в соответствии с вариантом индивидуального задания. «Застройка» выполняется путем редактирования файла Labyrinth.txt в папке симулятора. Система управления должна обеспечивать интерфейс, содержащий кнопку старта и остановки управляющей программы, а также текстового окна для вывода сообщений о ходе работы и возникающих ошибках.

На рисунке 1 - рабочая зона.

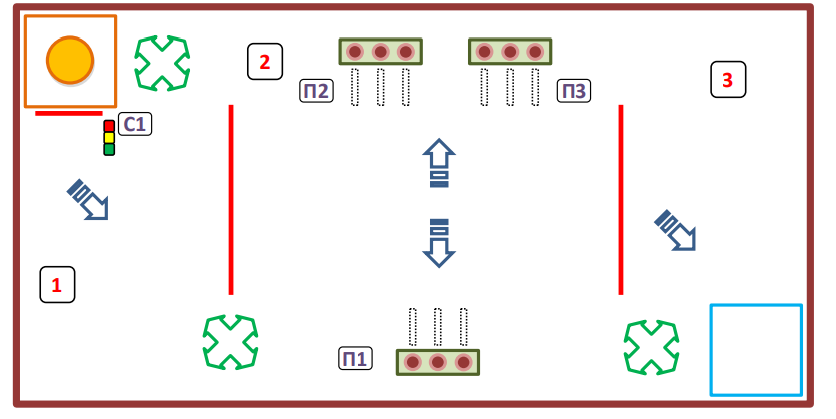
**Индивидуальное задание**

Рисунок 1. Рабочая зона.

Задание Варианта 5.

Участникам необходимо составить и реализовать алгоритм и программу движения робота, оснащённого логистическим захватом, чтобы он в автономном режиме выполнил перенос заготовок между логистическими платформами по следующему сценарию:

• Робот устанавливается в стартовую позицию. Светофор № 1 переключен на красный свет. Робот начинает движение со стартовой позиции по зелёному сигналу светофора № 1 (С1). Стартовая ориентация робота – на усмотрение команды.

• Робот выполняет движение по сектору № 1 в сектор № 2.

• Робот перемещается к логистической подставке № 1 (П1).

• Анализируя размещение заготовок, робот выбирает и захватывает заготовку красного либо синего цвета (заготовки будут установлены судьей в заранее неизвестные ячейки непосредственно после размещения робота на старте).

• Робот движется по сектору № 2 к логистической подставке № 2 (П2), если у него заготовка красного цвета, либо к логистической подставке № 3 (П3), если у него заготовка синего цвета.

• Достигнув соответствующей логистической подставки, Робот сканирует размещение заготовок на логистической подставке для поиска свободного места и устанавливает туда привезённую заготовку.

• Затем робот возвращается к логистической подставке № 1 (П1). И повторяет операцию сканирования и доставки заготовки.

• Если все заготовки перенесены с логистической подставки № 1 (П1), то робот движется к зоне финиша.

• После заезда в зону финиша робот останавливается внутри этой зоны. Навигация робота может осуществляться при помощи только технического зрения, индуктивных и оптических датчиков. После доставки последней заготовки Робот не обязан возвращаться к логистической подставке № 1 (П1). Участники должны реализовать алгоритм допускающий возможность размещения на логистической подставке менее трех заготовок.

**Этап моделирования**

1. При анализе программы была разделена на состояния. Было выделено 4 основных состояния, таких как ожидания подсоединения, ожидание запуска отправки сообщений для старта движения робота, аварийная ситуация, из-за которой работа программы останавливалась и состояние, при котором программа закрывалась, по желанию пользователя. На рисунке 2 отображена диаграмма состояний верхнего уровня программы.

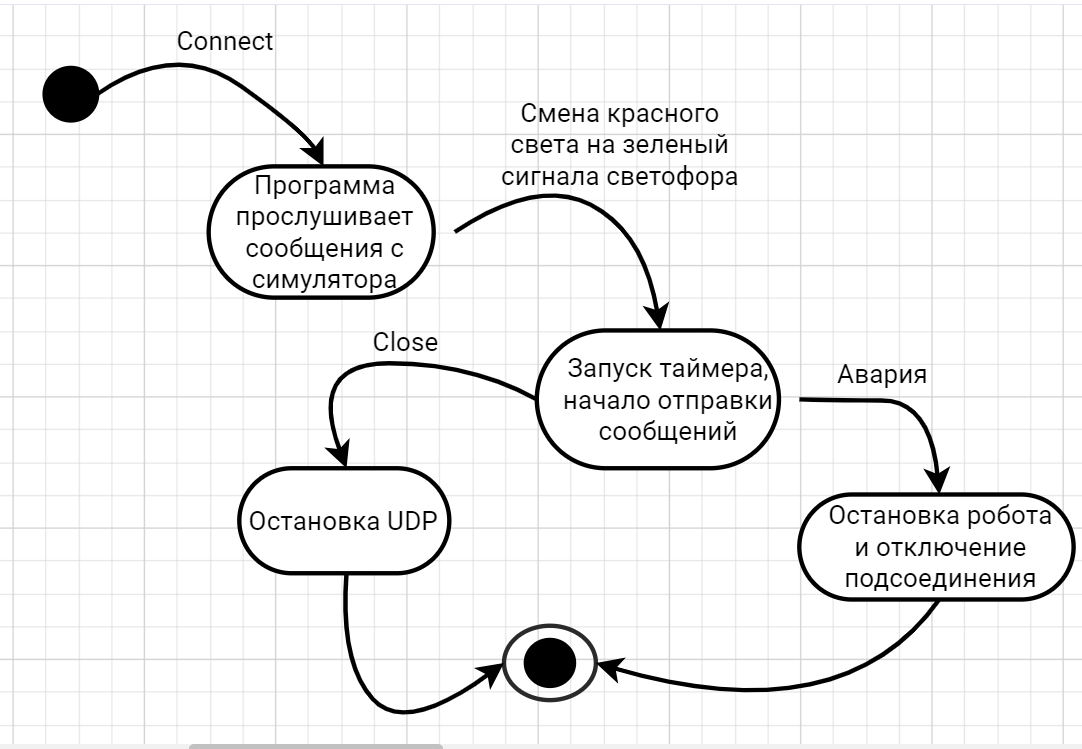
****

Рисунок 2 – Диаграмма состояний верхнего уровня

1. Необходимо было продумать состояния, в которых будет находится робот для отправки корректных значений для его действий. Опираясь на задание, робот может находится в 3 рабочих зонах, разделенных линиями. Так состояния поделились на зону старта, рабочую зону и зону финиша. В этих состояниях есть еще под состояния.
2. В зоне старта основная задача доехать до линии к 2 зоне. Основные состояния робота буду выезд с зоны старта и движение к линии (Рисунок 3).

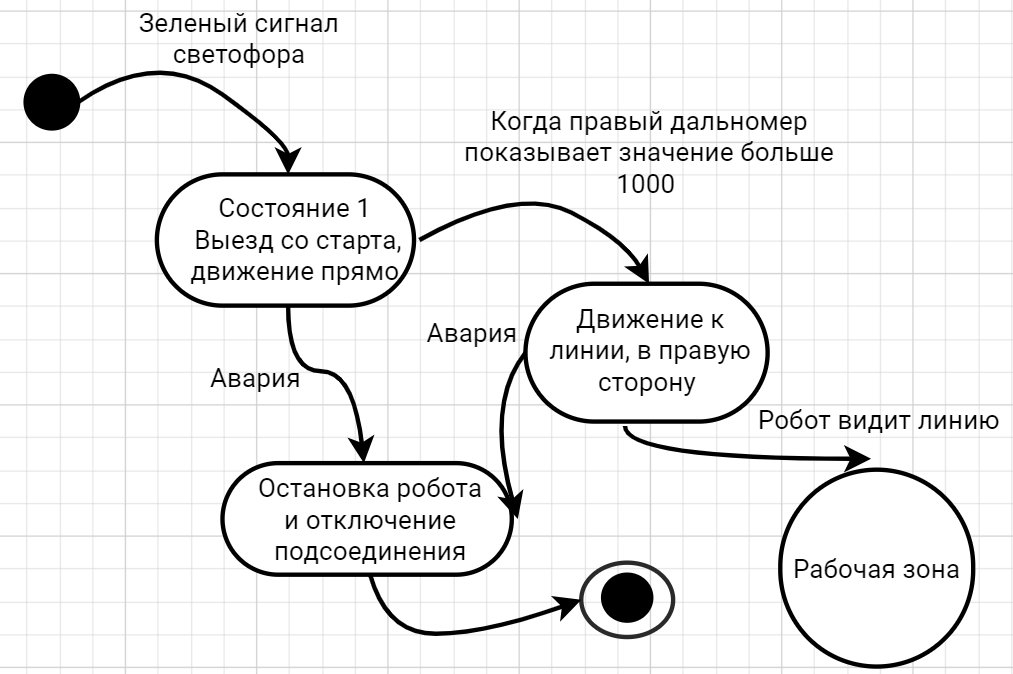
****

Рисунок 3 – Блоки состояний зоны старта

1. В рабочей зоне задачи робота: подъезд к подставкам, определение количество и цвет заготовок и взять одну, либо поставить. Всего 3 подставки, все берутся заготовки из 1 подставки, в зависимости от цвета заготовки ставятся в 2 или 3 подставку. Для начала, когда робот попадает в рабочую зону, ему необходимо подъехать к 1 подставке, остановится определить цвет, количество и взять банку. Далее доехать до подставок 2 или 3, там оставить заготовку и вернуться. То есть у робота будут состояние: движение к подставке 1, остановка (в которой робот берет или ставит заготовку, а если это подставка 1, то еще и определяет количество оставшихся, а также цвет заготовки, которую взял), движение к следующей подставки, движение к линии в зону 3 (Рисунок 4).

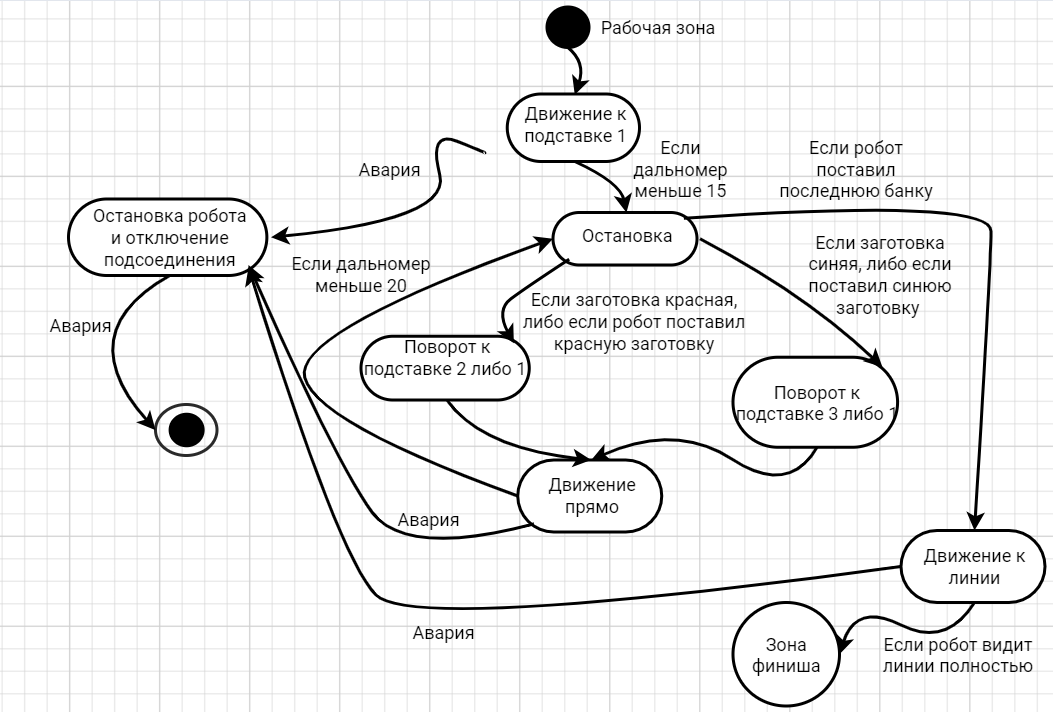
****

Рисунок 4 – Блоки состояний в рабочей зоне.

1. В момент, когда робот будет находится в состоянии остановки, поворотов будут еще использоваться под состояния для определения направления и дальнейших переключений состояний (Рисунок 5 и 6). Состояния 4 и 5 поворотов к подставкам одинаковы, различны только углы поворотов, поэтому на рисунке 6 будет отображен только под состояния поворота к подставке 2.

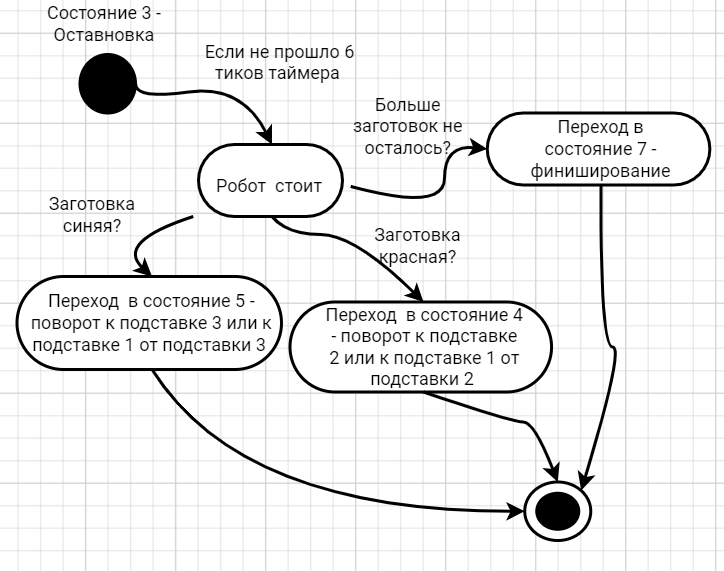
****

Рисунок 5 – Под состояние состояния 3 – Остановки.

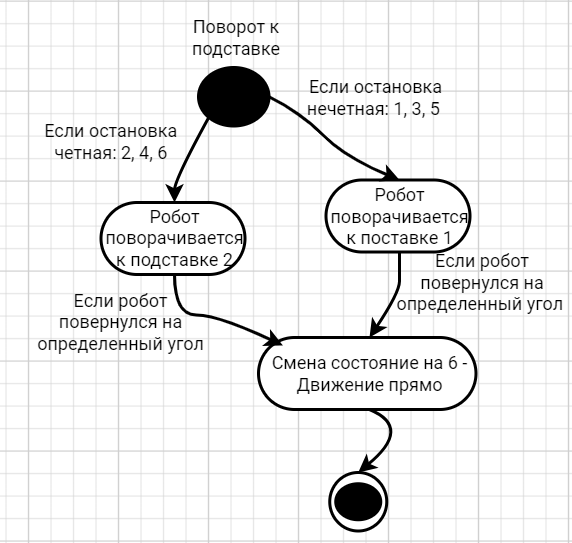


Рисунок 6 – Под состояние состояния 4 – Поворот к подставке

1. В зоне финиша робот имеет одно состояния, но несколько под состояний. Они включают в себя выезд с линии, движение к зоне финиша и парковка, далее робот останавливается (Рисунок 7).

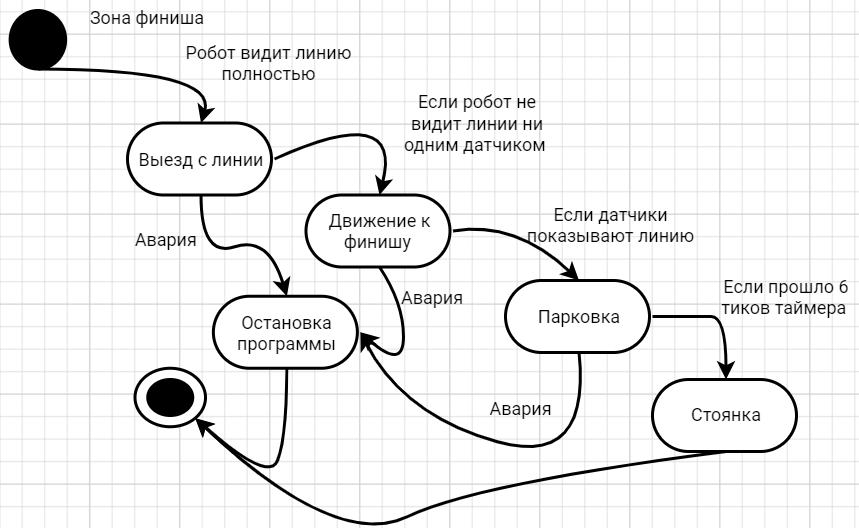
****

Рисунок 7 – Блок состояний зоны финиша

**Этап разработки**

1. Было создано приложении Win Form’s в котором отображались порты локальные и сервера, ip-адрес, сообщения, которые приходили на сервер, состояния робота, кнопки для светофора, остановки программы и подсоединения.
2. В коде было реализовано подсоединение по протоколу UDP.
3. Была реализована обработка полученных данных и система отправки сообщений. Отправка происходила по тикам таймера.
4. Был разработан алгоритм, по которому робот мог переходить в различные состояния в зависимости от факторов, в каких он находится.
5. В зависимости от состояния, которое можно было отследить, через алгоритм, происходил вывод сообщений состояния робота для пользователя в ListBox.
6. Так как не известно, сколько заготовок и какого они цвета, реализован рандомный подбор значений.
7. Реализована остановка робота при столкновение его со стенкой. Это первое условие, которое проверяется всегда.
8. В результате программа реализовывает автоматическое движение роботы. Пользователь управляет стартом, и остановкой программы, также светофором, который в начальный момент времени красный, и отслеживает состояния робота.

**Вывод**

Были разработаны диаграммы состояний, которые помогли понять устройство кода при реализации приложения. Приложение было разработано под диаграммы, при этом возникли некоторые переработки задумки. В конечном варианте программы отсутствует явное разделение на зоны старта, рабочую и финиша. Все состояния прописаны в таймере, где и происходит вся работа логики автоматизированного управления роботом.

**Приложение**

Листинг 1. Код программы.

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.Net;

using System.Net.Sockets;

using System.Threading;

using System.Timers;

using System.Runtime.InteropServices.ComTypes;

using System.Text.RegularExpressions;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Rebar;

namespace Laba2

{

public partial class Form1 : Form

{

public delegate void ShowMessage(string message);

public ShowMessage myDelegate;

UdpClient udpClient;

Thread thread;

bool isClientOpen = false;

int svet = 0;

int nm = -1;

int fm = 70;

int bm = 0;

int co = 0;

int dm0 = 0;

int dm7 = 0;

public int n;

public int s;

public int c;

public int le;

public int re;

public int az;

public int b;

public int d0, d1, d2, d3, d4, d5, d6, d7;

public int l0, l1, l2, l3, l4;

int tik = 0;

private Random random = new Random();

private int count;

private int[] color;

private int ostanovka;

int r = -1;

int l = 0;

int g = 0;

Dictionary<string, int> dataR = new Dictionary<string, int>

{

{ "n", 0 },

{ "s", 0 },

{ "c", 0 },

{ "le", 0 },

{ "re", 0 },

{ "az", 0 },

{ "b", 0 },

{ "d0", 0 },

{ "d1", 0 },

{ "d2", 0 },

{ "d3", 0 },

{ "d4", 0 },

{ "d5", 0 },

{ "d6", 0 },

{ "d7", 0 },

{ "l0", 0 },

{ "l1", 0 },

{ "l2", 0 },

{ "l3", 0 },

{ "l4", 0 },

};

public Form1()

{

InitializeComponent();

txtStartIP.Text = "127.0.0.1";

Svetofor.BackColor = Color.Red;

Svetofor.Visible = false;

}

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

{

myDelegate = new ShowMessage(ShowMessageMethod);

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Escape)

{

thread.Abort();

udpClient.Close();

Close();

}

}

private void ReceiveMessage()

{

if (isClientOpen)

{

while (true)

{

IPEndPoint remoteIPEndpoint = new IPEndPoint(IPAddress.Any, Int32.Parse(myPort.Text));

byte[] content = udpClient.Receive(ref remoteIPEndpoint);

if (content.Length > 0)

{

string message = Encoding.ASCII.GetString(content);

this.Invoke(myDelegate, new object[] { message });

if (message.Length > 50)

{

// При выборе новой строки в ListBox вызываем обработку строки

string selectedString = message;

DataRecieve(selectedString);

}

}

}

}

}

public void DataRecieve(string message)

{

string[] dataEx = Regex.Split(message, @"\D+");

n = int.Parse(dataEx[1]);

s = int.Parse(dataEx[2]);

c = int.Parse(dataEx[3]);

le = int.Parse(dataEx[4]);

re = int.Parse(dataEx[5]);

az = int.Parse(dataEx[6]);

b = int.Parse(dataEx[7]);

d0 = int.Parse(dataEx[9]);

d1 = int.Parse(dataEx[11]);

d2 = int.Parse(dataEx[13]);

d3 = int.Parse(dataEx[15]);

d4 = int.Parse(dataEx[17]);

d5 = int.Parse(dataEx[19]);

d6 = int.Parse(dataEx[21]);

d7 = int.Parse(dataEx[23]);

l0 = int.Parse(dataEx[25]);

l1 = int.Parse(dataEx[27]);

l2 = int.Parse(dataEx[29]);

l3 = int.Parse(dataEx[31]);

l4 = int.Parse(dataEx[33]);

//return new[] { n, s, c, le, re, az, b, d0, d1, d2, d3, d4, d5, d6, d7, l0, l1, l2, l3, l4 };

dataR = new Dictionary<string, int>

{

{ "n", n },

{ "s", s },

{ "c", c },

{ "le", le },

{ "re", re },

{ "az", az },

{ "b", b },

{ "d0", d0 },

{ "d1", d1 },

{ "d2", d2 },

{ "d3", d3 },

{ "d4", d4 },

{ "d5", d5 },

{ "d6", d6 },

{ "d7", d7 },

{ "l0", l0 },

{ "l1", l1 },

{ "l2", l2 },

{ "l3", l3 },

{ "l4", l4 },

};

}

private void ShowMessageMethod (string message)

{

richText.Items.Add(message);

}

private void ShowSostoiania(string write)

{

sostoiania.Items.Add(write);

}

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

{

thread.Abort();

udpClient.Close();

BtnClose.Visible = false;

isClientOpen = false;

}

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

{

try

{

udpClient = new UdpClient(int.Parse(myPort.Text));

BtnClose.Visible = true;

Svetofor.Visible = true;

isClientOpen = true;

thread = new Thread(new ThreadStart(ReceiveMessage));

thread.IsBackground = true;

thread.Start();

string write = "Робот готов к движению, красный свет";

ShowSostoiania(write);

}

catch

{

MessageBox.Show("Connect error");

}

}

private void SendUDPMessage(string s)

{

if (udpClient != null)

{

Int32 port = int.Parse(portClient.Text);

IPAddress ip = IPAddress.Parse(txtStartIP.Text.Trim());

IPEndPoint ipEndPoint = new IPEndPoint(ip, port);

byte[] content = Encoding.ASCII.GetBytes(s);

try

{

int count = udpClient.Send(content, content.Length, ipEndPoint);

if (count > 0)

{

ShowMessageMethod("Message has been sent.");

}

}

catch

{

ShowMessageMethod("Error occurs.");

}

}

}

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

{

if (svet == 0)

{

Svetofor.BackColor = Color.Green;

Svetofor.Text = "Green";

// Устанавливаем интервал в миллисекундах (в данном случае 200 мс)

myTimer.Interval = 200;

count = random.Next(1, 4);

color = new int[] { random.Next(0, 2), random.Next(0, 2), random.Next(0, 2) };

ostanovka = 0;

// Запускаем таймер

myTimer.Start();

string write = "Свет зеленый, начало движения";

ShowSostoiania(write);

svet = 1;

}

}

private void MoveRobot (int co)

{

if (co == 1)

{

string write = "Робот находится в 1 секторе, едет ко 2";

ShowSostoiania(write);

}

else if (co == 2)

{

string write = "Робот пересек границу 2 сектора, едет к подставке 1 для заготовок";

ShowSostoiania(write);

}

else if (co == 3)

{

if (ostanovka == 1 || ostanovka == 3 || ostanovka == 5)

{

string write = "Робот берет заготовку с подставки 1";

ShowSostoiania(write);

}

else if (ostanovka == 2 || ostanovka == 4 || ostanovka == 6)

{

if (color[r] == 0)

{

string write = "Робот поставил заготовку в подставку 2";

ShowSostoiania(write);

}

else

{

string write = "Робот поставил заготовку в подставку 3";

ShowSostoiania(write);

}

}

}

else if (co == 4)

{

string write = "Робот взял красную заготовок, осталось зоготовок" + count;

ShowSostoiania(write);

}

else if (co ==5)

{

string write = "Робот взял синию заготовок, осталось заготовок" + count;

ShowSostoiania(write);

}

else if (co == 7)

{

string write = "Робот пересек границу 3 сектора, едет к финишу";

ShowSostoiania(write);

}

else if (co == 8)

{

string write = "Робот финишировал";

ShowSostoiania(write);

}

else if (co == 9)

{

string write = "Ошибка! Робот врезался. Запустите программу заново!";

ShowSostoiania(write);

}

}

private void SendValues(int nm, int fm, int bm)

{

//string s = MyMessage.Text;

string s = "{ \"N\":" + nm +", \"M\":0, \"F\":"+ fm +", \"B\":" + bm + ", \"T\":0}\n";

SendUDPMessage(s);

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (b == 1)//авария

{

nm += 1;

bm = 0;

fm = 0;

co = 9;

MoveRobot(co);

thread.Abort();

udpClient.Close();

myTimer.Stop();

}

else if (co==0)//выезжание со старта

{

nm += 1;

if (d2 > 1000)

{

co = 1;

MoveRobot(co);

}

}

else if (co == 1)//движение к рабочей зоне

{

nm+= 1;

bm = -4;

if (l0 == 0 || l1==0 || l2==0 || l3 == 0 || l4 == 0)

{

co = 2;

MoveRobot(co);

}

}

else if (co == 2)//движение к подставке 1

{

nm += 1;

bm = 4;

fm = 75;

if (d0<=20)

{

co = 3;

MoveRobot(co);

}

}

else if (co == 3)//остановка

{

if (tik < 6)

{

nm += 1;

bm = 0;

fm = 0;

tik += 1;

if (tik == 1)

{

ostanovka += 1;

MoveRobot(co);

if (ostanovka == 1 || ostanovka == 3 || ostanovka == 5)//если у подставке 1

{

r += 1;

count -= 1;

l = 0;

g = 0;

}

}

}

else

{

if (count == 0 && (ostanovka == 2 || ostanovka == 4 || ostanovka == 6))// если больше не осталось заготовок

{

co = 7;

MoveRobot(co);

}

else if (color[r]==0) //если заготовка красная

{

co = 4;

MoveRobot(co);

}

else//если заготовка синяя

{

co = 5;

MoveRobot(co);

}

}

}

else if (co == 4)

{

if (az <= 258 && (ostanovka == 1 || ostanovka == 3 || ostanovka == 5) && g == 0) //поворот к подставке 2

{

nm += 1;

bm = 20;

fm = 0;

}

else if (az > 47 && (ostanovka == 2 || ostanovka == 4 || ostanovka == 6)) //подставке к подставке 1

{

nm += 1;

bm = -20;

fm = 0;

}

else//движение прямо

{

co = 6;

g = 1;

}

}

else if (co == 5)

{

if (az <= 278 && (ostanovka == 1 || ostanovka == 3 || ostanovka == 5) && g == 0) //поворот к подставке 3

{

nm += 1;

bm = 20;

fm = 0;

}

else if (az > 47 && (ostanovka == 2 || ostanovka == 4 || ostanovka == 6)) //порворот к подставке 1

{

nm += 1;

bm = -20;

fm = 0;

}

else//движение прямо

{

co = 6;

g = 1;

}

}

else if (co == 6)

{

if (d0 > 15)//движение прямо

{

nm += 1;

bm = 0;

fm = 75;

}

else if (d0 <= 20)//остановка

{

co = 3;

tik = 0;

MoveRobot(co);

}

}

else if (co == 7)

{

if (az > 5 && l == 0)//поворот в сторону движения к финишу

{

nm += 1;

bm = -20;

fm = 0;

}

else if (az < 5 && (l0 != 0 && l1 != 0 && l2 != 0 && l3 != 0 && l4 != 0)) //движение к линии

{

l = 1;

nm += 1;

bm = 0;

fm = 100;

}

else if ((l0 == 0 || l1 == 0 || l2 == 0 || l3 == 0 || l4 == 0) && l==1)//если робот заметил линию

{

if ((l0 != 0 || l1 != 0 || l2 != 0 || l3 != 0 || l4 != 0))//выравнивание робота по линии

{

nm += 1;

bm = -10;

fm = 20;

}

else if ((l0 == 0 && l1 == 0 && l2 == 0 && l3 == 0 && l4 == 0))//смена статуса

{

l = 2;

co = 8;

MoveRobot(co);

}

}

}

else if (co == 8)

{

if ((l0 == 0 && l1 == 0 && l2 == 0 && l3 == 0 && l4 == 0) && l==0 && tik >=6) //остановка

{

nm += 1;

bm = 0;

fm = 0;

}

else if ((l0 == 0 && l1 == 0 && l2 == 0 && l3 == 0 && l4 == 0) && l == 0) //парковка

{

nm += 1;

bm = 10;

fm = 40;

tik ++;

}

else if ((l0 == 0 && l1 == 0 && l2 == 0 && l3 == 0 && l4 == 0) && l == 2) //выезжает с линии

{

nm += 1;

bm = 0;

fm = 100;

tik = 0;

}

else//движение к финишу

{

nm += 1;

bm = 5;

fm = 100;

l = 0;

}

}

SendValues(nm, fm, bm);

}

}

}

//{"N":1, "M":0, "F":50, "B":10, "T":0}