МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ автономное ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



Лабораторная работа № 2:

«Наивные (примитивные) алгоритмы управления движением мобильного робота»

По дисциплине: «Задачи планирования движения и навигация в робототехнике»

Группа 221-328

Студент Шурова Д.С.

Дата 05.11.2024

Преподаватель Идиатулов Т.Т

2024

**Задание (общее)**

На застроенной в соответствии с индивидуальным заданием рабочей зоне обеспечить перемещение робота и поиск объектов в рамках указанного индивидуального задания.

Разработать на языке C# (WinForms) приложение-сервер, реализующий функции системы управления роботом с обеспечением визуализации мониторинговых данных, определенных индивидуальным заданием. Обмен данными с симулятором должен выполняться по протоколу UDP с передачей строк в формате JSON.

В начале работ реализовать функциональность UDP сервера (например, через класс udpClient) и отображение пришедших сообщений в listBox. Убедиться в получении сообщений от симулятора.

В рамках приложения реализовать интерфейс для ввода параметров команд управления с помощью текстовых полей и ползунковых регуляторов, а также последующей отправки по нажатию отдельных кнопок. Сборку JSON строки рекомендуется выполнять без использования дополнительных библиотек.

Затем реализовать парсинг полученных от симулятора сообщений с выводом значений в отдельные текстовые поля (textBox’ы).

Следующим шагом необходимо разместить в приложении таймер для реализации режима автоматического движения и обеспечить управление его работой. После чего в обработчике таймера нужно сформировать код управления мобильным роботом, проверяющим значения из текстовых полей с данными мониторинга, и затем формирующим и отправляющим управляющие команды.

Индивидуальные варианты указывают на то, какой из базовых алгоритмов движения необходимо реализовать, а также дают указания на вид трассы.

При тестировании полученных алгоритмов схему поля (лабиринта) рекомендуется изменить. Также преподаватель может предложить другой вариант схемы поля в качестве индивидуального варианта, отличный от указанного в данном тексте задания.

**Вариант 3** (Движение по линии)

Используется робот типа 0 (круглый, дифференциальный привод). Реализовать движение по кольцевой трассе по часовой стрелке по размеченной линии (использовать смарт-камеру для детекции линии), обозначенной кодом 1 на поле. На старте выполнять движение алгоритмом случайного блуждания с равными интервалами до нахождения линии, на линии развернуться влево и выполнять движение по линии.

**Ход работы**

1. Разработано приложение на языке C# (WinForms), которое будет реализовывать функции управления мобильным роботом через UDP-протокол. Приложение обмениваться данными с симулятором, отправляет JSON команды на перемещение робота.
2. На первом этапе было реализовано приложение, содержащее базовый функционал UDP-сервера. Используя класс UdpClient, было создано подключение для приема сообщений от симулятора. Каждое полученное сообщение отображалось на форме приложениям в Message. Для асинхронного получения сообщений использовался метод ReceiveAsync.
3. Для управления роботом на форме приложения были добавлены текстовые поля и кнопки, позволяющие вводить значения перемещения для энкодера для управления роботом. Команды отправлялись симулятору в формате JSON с помощью метода отправки данных через UDP.
4. Если пользователь выбирал автоматический режим управления, то в методе timer1\_Tick робот выполняет задачи последовательно.
5. Изначально в режиме блуждания, робот движется по 3 направлениям рандомно (вперед по часовой стрелки, вперед против часовой стрелки, прямо). Также в этих режимах рандомно выставляется скорость для движения или поворота, что позволяет роботу постоянно менять свою траекторию. В случае, столкновения со стенкой, робот движется назад и обратной скоростью поворота, что позволяет ему ехать в противоположном направлении. Но, если робот врежется задней частью корпуса, то движение назад не поможет ему выйти из аварии. Для этого, после попытки движения в противоположном направлении, если такой вариант не срабатывает, программа отправляется команду движения вперед, что позволяет роботу отъехать от стенки и продолжить рандомное движение.
6. Когда робот находит линии с помощью smart-камеры, то начинает разворот влево для движения по часовому направлению ориентируясь по азимуту. Далее начинается движение по линии используя датчики освещенности. После выполнения блуждания, а также после выравнивания для движения по часовому направлению сменяются флаги, которые не позволяют вернуться к пройденным алгоритмам.
7. Движение по линии делится на несколько состояний. Робот движется прямо в случае, если центральный датчик освещенности равен 0. Иначе, проверяются крайнее датчики, чтобы робот смог повернутся в сторону линии и найти линию датчиком l0(центральным). (Рисунок 1)

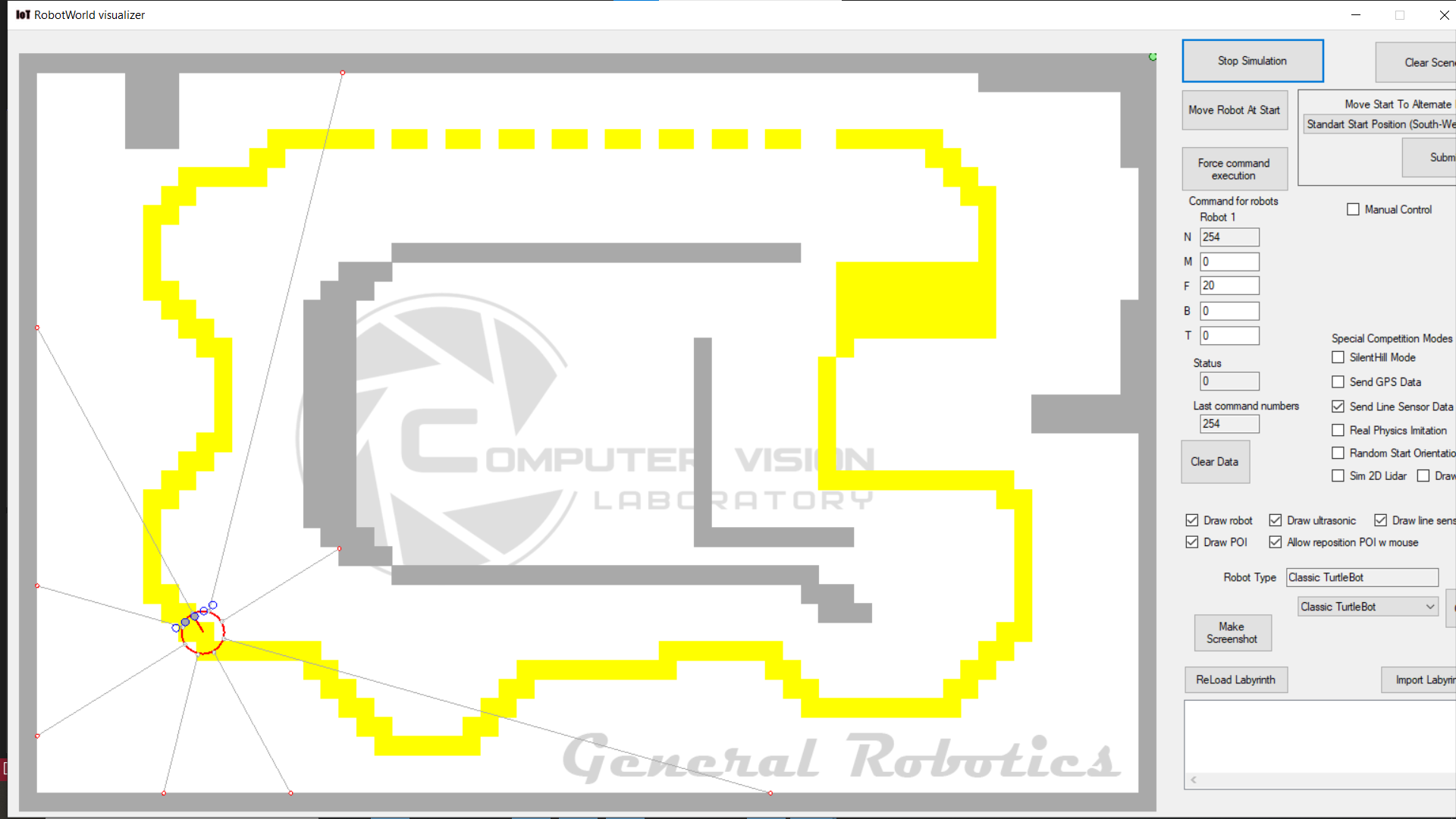


Рисунок 1 – Фрагмент движение по линии

**Вывод**

Было разработано приложение для управления движением робота по линии в двух режимах, автоматический и ручной. Команды отправляются в симулятор по протоколу UDP в JSON формате. Программа обрабатывает значение датчиков, полученных с симулятора, чтобы отправить обратно команды.

**Приложение А**

Листинг 1. Основной код

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;

using System.Diagnostics;

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

using System.Xml.Linq;

namespace Laba2

{

public partial class Form1 : Form

{

public delegate void ShowMessage(string message);

public ShowMessage myDelegate;

UdpClient udpClient;

Thread thread;

bool isClientOpen = false;

int nm = -1;//постоянно увеличивает номер команды

int fm = 70;//скорость движения

int bm = 0;//скорость поворота

int co = 0;//для автоматического режима переходов

int bo = 0;

int lenc = 0;

int l = 0;

int tic = 0;

int t = 0;//счетчик для таймера в блуждании

public int n, s;

public int le;

public int re;

public int b;

public int az;

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

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

int c;

int number = 0;

int metr;

int rand = 100; //блуждание рандомный номер команды

Boolean flag = true;//когда камера найдет линию, то переключится

Boolean flag1 = true;// когда робот развернется для движения по часовой, то переключится

Boolean flag2 = true;// чтобы перейти на движение состояние по линии

int angle;//запоминает скорость поворта в режиме блуждания

Random random = new Random();

/\*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";

groupBox1.Visible = false;

textBox1.Text = "0";

}

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]);

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 },

{ "le", le },

{ "re", re },

{ "b", b },

{ "l0", l0 },

{ "l1", l1 },

{ "l2", l2 },

{ "l3", l3 },

{ "l4", l4 },

};\*/

}

private void ShowMessageMethod (string message)

{

richText.Items.Insert(0, $"n= {n} s= {s} c= {c} b= {b} d0= {d0} d1= {d1} d2= {d2} d3= {d3} le= {le} re= {re} b= {b} l0={l0} l1={l1} l2={l2} l3={l3} l4={l4} ");

}

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

{

thread.Abort();

udpClient.Close();

BtnClose.Visible = false;

isClientOpen = false;

}

private void button4\_Click(object sender, EventArgs e)

{

number = 1;//forward

metr = int.Parse(textBox1.Text);

}

private void button3\_Click(object sender, EventArgs e)

{

number = 2;//back

metr = int.Parse(textBox1.Text);

}

private void button1\_Click(object sender, EventArgs e)

{

number = 3;//left

metr = int.Parse(textBox1.Text);

}

private void button2\_Click(object sender, EventArgs e)

{

number = 4;//right

metr = int.Parse(textBox1.Text);

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (!checkBox1.Checked)

{

groupBox1.Visible = true;

}

else { groupBox1.Visible = false; }

}

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

{

try

{

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

BtnClose.Visible = true;

isClientOpen = true;

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

thread.IsBackground = true;

thread.Start();

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

myTimer.Interval = 200;

// Запускаем таймер

myTimer.Start();

checkBox1.Visible = false;

}

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 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)

{

nm += 1;

if (checkBox1.Checked)

{

if (c != 1 && flag)

{

if (t == 0 || b==1 && flag1)

{

t = 15;

rand = random.Next(0, 3);

}

if (rand == 0 && b != 1)

{

fm = 50;

bm = 30;// random.Next(0, 50);

t -= 1;

angle = bm;//запоминает угол

flag1 = true;

}

else if (rand == 1 && b != 1)

{

fm = 50;

bm = -30; //random.Next(-50,0);

t -= 1;

angle = bm;

flag1 = true;

}

else if (rand == 2 && b != 1)

{

fm = 50;

bm = 0; //random.Next(-50,0);

t -= 1;

angle = bm;

flag1 = true;

}

else if (b==1 || t!=0)

{

if (t == 15)

{

flag1 = false;

}

t -= 1;

fm = -30;

bm = angle \* (-1);// в случае столкновения будет двигаться в противоположное направление

rand = 100;

if (b == 1 && t == 0)//в случае, если он не может сдать назад

{

fm = 100;

bm = 0;

}

}

}

else if ((c==1 || !flag) && flag2)// настройка для движения по часовой

{

flag = false;

if (az >0 && az < 180 && l0 == 0)

{

flag2 = false;

}

else if (az > 180 || l0==100)

{

fm = 0;

bm = 40;

}

}

else if (!flag2)

{

if (l0 == 0)

{

fm = 20;

bm = 0;

}

else if (l3 == 0 && (l0 == 100 || l1 == 100 || l2 ==100))

{

fm = 0;

bm = 10;

}

else if (l4 == 0 && (l0 == 100 || l1==100 || l2 ==100))

{

fm = 0;

bm = -10;

}

else if (l4 == 100 && l3 == 100 && l2 == 100 && l1 == 100 && l0 == 100)

{

fm = 20;

bm = 0;

}

}

}

else

{

if (number == 1 && metr > le && metr!=null)//forward

{

fm = 70;

bm = 0;

}

else if (number == 2 && metr > le && metr != null)//back

{

fm = -70;

bm = 0;

}

else if (number == 3 && metr > re && metr != null)//left

{

fm = 0;

bm = 30;

}

else if (number == 4 && metr > le && metr != null)//right

{

fm = 0;

bm = -30;

}

else

{

fm = 0;

bm = 0;

}

}

SendValues(nm, fm, bm);

}

}

}

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