**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Организация графических систем  
Лабораторная работа №4

«Реализация процедур аудиозахвата (audio capture) или видеозахвата (video capture) в реальном времени»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | | | | |  | |  |  | | Янин С.А. | | |  |
| Группа |  | М-АС-19 | | | |  | | |  | |  |
|  | | |  |  | | | |  |  | | | |  |
| Руководитель | | |  |  | | | |  | Кургасов В.В. | | | |  |

Липецк 2020 г.

**Задание**

Реализовать обработку аудио или видеопотока с использованием вычислительных возможностей видеоадаптера. Наглядно продемонстрировать алгоритмы обработки аудио или видеопотоков.

# Оглавление

Оглавление 3

Введение 4

1 Теоретические сведения 6

1.1 MediaCapture 6

1.2 SetInterval 6

1.3 Speech Recognition 7

1.4 Язык реализации 8

2 Исходный код 10

3 Результаты работы программы 12

Вывод 14

# Введение

С момента появления первых ЭВМ одним из наиболее важных вопросов развития компьютерной техники был процесс взаимодействия человека с машиной. Долгое время это было доступно только узким специалистам – технологи «общались» с машиной через посредника-программиста. Такая ситуация просуществовала вплоть до появления диалогового интерфейса, когда пользователь смог лично вводить с клавиатуры адресованную машине команду и получать осмысленный ответ. Появление графического интерфейса, при котором отпала необходимость в знании человеком каких-либо команд, привела к повсеместному распространению персональных компьютеров.

Однако человек всегда стремился к более универсальному и естественному способу взаимодействия с ЭВМ. Еще в эпоху перфокарт в научно-фантастических романах человек разговаривал с компьютером, как с равным себе. Тогда же были предприняты первые шаги по реализации речевого интерфейса.

Тем не менее, если сравнить показатели современных систем распознавания с показателями систем времен начала зарождения этой области науки, то можно сказать, что за прошедшие десятки лет исследователи недалеко продвинулись. Это заставляет некоторых специалистов сомневаться относительно возможности реализации речевого интерфейса в ближайшем будущем. Другие считают, что задача уже практически решена. Большинство экспертов сходится во мнении, что для развития распознавания речи потребуется какое-то время.

В настоящее время научное сообщество вкладывает гигантское количество денег в развитие know-how и научно-исследовательские разработки для решения проблем автоматического распознавания и понимания речи. Это стимулируется практическими требованиями, связанными с созданием системы военного и коммерческого назначения. Не касаясь первого из них, можно указать, что только в европейском сообществе объем продаж систем гражданского назначения составляет несколько миллиардов долларов. При этом следует обратить внимание на то, что в практическом использовании отсутствуют системы, считающиеся по непонятным причинам вершиной развития систем автоматического распознавания речи. Это системы, которые можно назвать демонстрационными и которые 50 лет назад назывались «фонетическими печатающими машинками». Их целью является перевод речи в соответствующий письменный текст.

# 1 Теоретические сведения

# 1.1 ****MediaCapture****

Класс MediaCapture предоставляет широкий набор API, которые обеспечивают низкоуровневое управление конвейером захвата и поддерживают расширенные сценарии захвата, но цель этой статьи — помочь быстро и просто добавлять в приложение основные функции захвата мультимедиа

Метод getUserMedia() — отличная стартовая точка в изучении Media Capture API. Вызвов getUserMedia() принимает MediaStreamConstraints как аргумент, который определяет настройки и/или требования к устройствам захвата и захваченным медиапотокам, такие как громкость микрофона, разрешение видео, какая камера включена. Через MediaStreamConstraints, вы так же можете использовать определенное устройство захвата, используя его ID, который может быть получен через enumerateDevices() метод. Когда пользователь дает разрешение, getUserMedia() может вернуть обещание вместе с MediaSteam объектом, если определенный MediaStreamConstraints будет найден.

Если вы хотите просто захватить фотографию или видео и не планируете добавлять дополнительные возможности захвата мультимедиа или не хотите создать собственный пользовательский интерфейс камеры, можно использовать класс CameraCaptureUI, который позволяет просто запустить встроенное приложение камеры Windows и получить созданную фотографию или видео.

# 1.2 SetInterval

Функция setInterval() неоднократно вызывает функцию или выполняет переданный код с указанной временной задержкой между каждым вызовом.

Примечание: данная функция вызывается на исполнение до тех пор, пока не будет закрыто окно с веб-страницей или вызвана функция clearInterval(), прерывающая работу setInterval(). Если нужно выполнение кода или вызов функции только однократно (с заданной задержкой), используйте функцию setTimeout().

Синтаксис

intervalID = setInterval(func[, delay, param1, param2, ...]);

intervalID = setInterval(code[, delay]);

Аргументы

func: Имя функции, которая будет вызываться после каждой задержки.

code: Строка, содержащая JavaScript-код, который будет выполняться после каждой задержки.

delay: Задержка, указываемая в миллисекундах, по истечении которой каждый раз будет выполняться вызов функции или переданный в виде строки код. Если указано значение меньше 10, то используется значение 10.

Примечание: 1 секунда = 1000 миллисекунд

param1, param2, ... : Параметры, которые будут переданы в качестве аргументов указанной функции.

Возвращаемое значение

Идентификатор: числовой идентификатор, который может быть использован функцией clearInterval().

# 1.3 Speech Recognition

Web Speech API позволяет веб-приложениям обрабатывать голосовые данные. Этот API состоит из двух компонентов:

Доступ к распознаванию речи осуществляется через интерфейс SpeechRecognition, который предоставляет возможность распознавать голосовой контекст из аудиовхода (обычно через службу распознавания речи устройства по умолчанию) и реагировать соответствующим образом. Для создания нового объекта SpeechRecognition используется конструктор интерфейса, который имеет ряд обработчиков событий, доступных для обнаружения, когда речь вводится через микрофон устройства. Интерфейс SpeechGrammar представляет собой контейнер для определенного набора грамматики, которую ваше приложение должно распознавать. Грамматика определяется с использованием формата грамматики JSpeech (JSGF).

Доступ к синтезу речи осуществляется через интерфейс SpeechSynthesis - компонент преобразования текста в речь, который позволяет программам считывать текст (обычно через синтезатор речи устройства по умолчанию). Различные типы голоса представлены объектами SpeechSynthesisVoice. Текст можно озвучить, передав его методу SpeechSynthesis.speak ().

# 1.4 Язык реализации

JavaScript — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален и дружелюбен к новичкам. Обладая большим опытом, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

* Программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов;
* Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook;
* Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

# 2 Исходный код

Файл «index.html»

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset**="utf-8"**>  
 <title>**Лабораторная №4**</title>  
 <script src**="js/app.js"**></script>  
 <link href**="css/app.css"** rel**="stylesheet"**>  
</head>  
<body>  
 <div class**="container"**>  
 <div id**="listen-button"** onclick**="**speech()**"**></div>  
 <textarea id**="text"** placeholder**="Нажмите на микрофон и произнесите что-нибудь. Здесь будет результат."**></textarea>  
 </div>  
</body>  
</html>

Файл «app.js»

//Объявляем экземпляр класса SpeechRecognition  
let *recognizer* = new webkitSpeechRecognition();  
//Переменная status используется для того, чтобы определить, запущен ли процесс распознавания речи, булево значение  
let *status* = false;  
/\*\*  
 \* Параметр, который обозначает прозрачность, по умолчанию непрозрачно, далее будет подставляться в свойства кнопки  
 \* алгоритм будет описан далее, это позволит кнопки "мигать", чтобы было понятно, что процесс распознования речи запущен  
 \*/  
let *opacity* = 1;  
  
// Оператор позволяет определить в какую сторону движется opacity, если -1, то вниз (тускнеет), иначе вверх  
let *operator* = -1;  
  
// Если указать это свойство как true, то будут возвращаться промежуточные результаты, что на мой взгляд удобно для пользователя  
*recognizer*.interimResults = true;  
// Этот параметр устанавливает обрабатываемый язык  
*recognizer*.lang = 'ru-Ru';  
  
//Здесь описан процесс "мигания" кнопки, всё помещается в setInterval, что позволяет постоянно обрабатывать описанный внутри код  
setInterval(() => {  
 //Если status == true, т.е. речь записывается для обработки, то меняем значение прозрачности, минимальный порог 0.3, чтобы кнопка совсем не пропала, максимальный 1  
 if(*status*){  
 if(*opacity* <= 0.3)  
 *operator* = 1;  
 if(*opacity* >= 1)  
 *operator* = -1;  
  
 *opacity* += 0.01 \* *operator*;  
 //Устанавливаем CSS свойство кнопки - opacity  
 *document*.getElementById('listen-button').style.opacity = *opacity*;  
 }  
 else //Если процесс распознавания речи не запущен, то делаем кнопку непрозрачной  
 *document*.getElementById('listen-button').style.opacity = 1;  
}, 10)  
  
  
/\*\*  
 \* onresult - свойство SpeechRecignition интерфейса, является обработчиком события, когда служба распозновния возвращает результат  
 \* а также промежуточные результаты (флаг выше)  
 \*/  
*recognizer*.onresult = function (*event*) {  
 //В event возвращается обработтаный объект, тут я выводил в консоли, чтобы узнать его структуру  
 *console*.log(*event*);  
 //В result складываем конечный результат, каждый раз возвращается полное предложение с самого начала  
 var result = *event*.results[*event*.resultIndex];  
 //устанавливаем значение для элемента textarea в html и передаем туда полученный текст  
 *document*.getElementById('text').value= result[0].transcript;  
  
 //Флаг isFinal говорит о том, запущен ли процесс распознавания речи, или он уже завершен. Если isFinal == true, это вернулся окончательный результат  
 if (result.isFinal) {  
 //Устанавливаю статус в false, для того, чтобы кнопка перестала "мигать"  
 *status* = false;  
 }  
};  
  
//Функция вызывается при нажатии на кнопку  
function speech () {  
 //Если процесс распознавания речи не был запушен, то запускаем, иначе прерываем, если, например, было повторное нажатие  
 if(!*status*) {  
 *recognizer*.start();  
 *status* = true;  
 }  
 else{  
 *recognizer*.stop();  
 *status* = false;  
 }  
  
}

# 3 Результаты работы программы

Для того чтобы проверить работу программы, необходимо открыть в любом браузере файл index.html. При запуске на странице браузера появится окно с кнопкой для начала обработки голоса и полем для отображения текста.

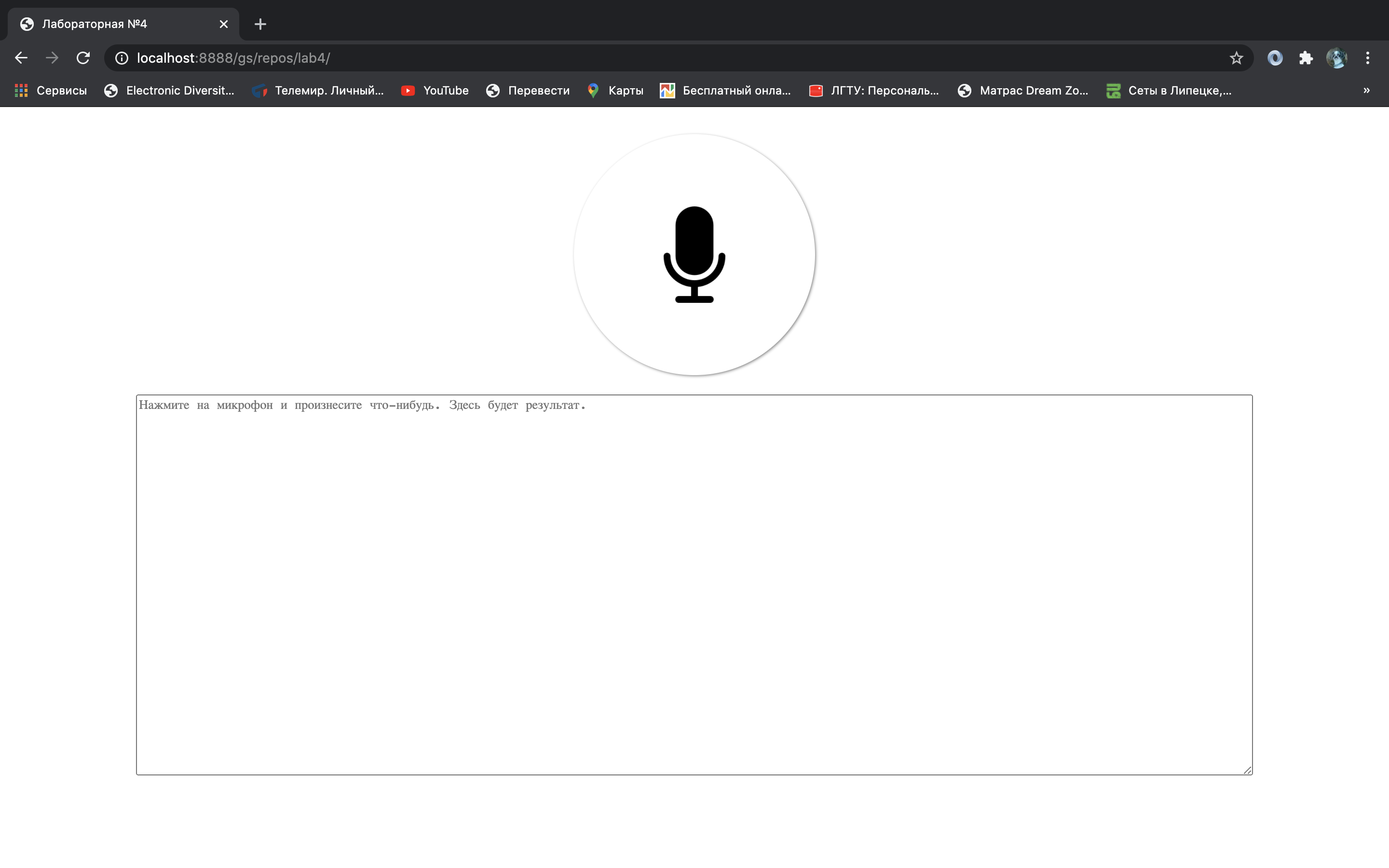


Рисунок 1 – Начальный экран

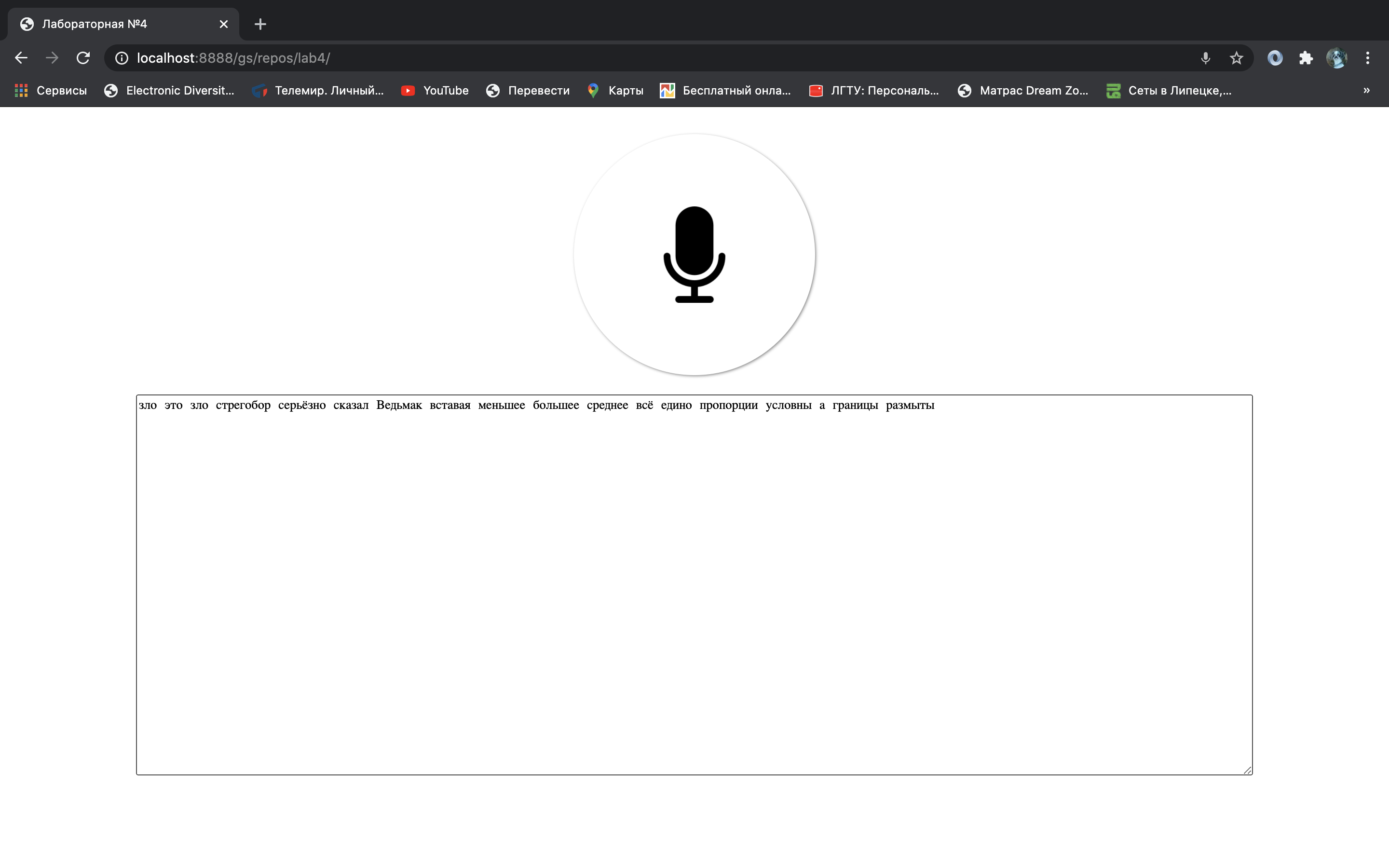


Рисунок 2 – Результат выполнения

# Вывод

В данной лабораторной работе мы ознакомились с библиотекой Speech Recognition. С помощью данной библиотеки мы смогли реализовать приложение, которое распознает речь.