Столыпинский вестник

Научная статья

Original article

УДК 004.428

РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА УПРАВЛЕНИЯ РОБОТОМ НА ОСНОВЕ ROS

WEB INTERFACE DEVELOPMENT ROBOT CONTROL BASED ON ROS

Стахеева Алина Алексеевна, магистрант, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск

Крайников Александр Николаевич, магистрант, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск

Staheeva A.A. staheeva.a@edu.narfu.ru

Krajnikov A.N. krajnikov.a@edu.narfu.ru

Аннотация

Данная работа посвящена разработке веб-сервиса, способного производить управление роботом, на базе фреймворка ROS. Разработан HTML код страницы, JavaScript код страницы, позволяющий с определённой периодичностью запрашивать значения из соответствующих топиков ROS, публиковать данные в топики в соответствии с нажатием кнопок на HTML, нажатием кнопок на клавиатуре или взаимодействием с джойстиком на веб-странице. Также рассматривается способ вывода потокового видео на веб-

страницу из топика ROS. По итогам работы была разработана концепция работы данного сервиса, были установлены необходимые пакеты ROS и произведена выгрузка веб-страницы в программу сервера.

Annotation

This work is devoted to the development of a web service capable of controlling a robot controlled by the ROS framework. An HTML page code, a JavaScript page code has been developed that allows you to request values from the corresponding ROS topics with a certain frequency, publish data to topics in accordance with pressing buttons on HTML, pressing buttons on the keyboard or interacting with a joystick on a web page. A method for outputting streaming video to a web page from a ROS topic is also considered. As a result of the work, the concept of this service was developed, the necessary ROS packages were installed, and the web page was uploaded to the server program.

Ключевые слова: ROS, Linux, ПО, система управления, веб-сервис.

Keywords: ROS, Linux, software, control system, web service.

Введение

Беспилотные дроны-почтальоны и беспилотные вездеходы-курьеры уже достаточно плотно укоренились в нашу жизнь. Однако даже полностью автоматизированные роботы требуют хотя бы минимальный интерфейс управления, например: кнопку «Включить-выключить». Но лучше все же если робот имеет полноценный интерфейс управления. С возможностью ручного пилотирования и отслеживания автоматического выполнения задач. Иначе каким образом можно будет экстренно остановить робота, если что-то пойдет не так, или выполнять с его помощью немного нестандартную задачу? Логическое умозаключения дает нам понять, что любой робот требует интерфейса управления.

В качестве базовой системы робота будет использоваться фреймворк ROS. ROS (Robot Operation System) - операционная система управления роботом. Данная система позволяет разрабатывать программное обеспечение

для роботов, комбинируя разнообразные модули. Модульная система, где каждая отдельная программа (нода) отвечает за отдельную функцию робота. Например, сбор данных с датчика расстояния. Данные ноды могут быть самостоятельно разработаны, а могут быть взяты из открытых источников. Так как общение между нодами стандартизировано и автоматизировано, можно не переживать за совместимость различных нод. [1]

В рамках разработки интерфейса управления также следует утвердить следующие параметры:

- интерфейс должен быть прост и понятен;
- интерфейс должен обладать всем необходимым функционалом для повседневной работы и для работы в нештатных ситуациях;
- интерфейс должен быть кроссплатформенным.

Таким образом были определены актуальность разработки веб-интерфейса и параметры системы.

Разработка концепции

В общем случае у нас должен быть доступный с любого устройства интерфейс, обладающий элементами управления, которые каким-то образом влияют на поведение робота.

В качестве основы интерфейса будем использовать веб-сервис доступный в локальной сети робота. Одним из основных преимуществ разработки именно веб-сервиса, это кроссплатформенность. Данный интерфейс можно будет использовать как на мобильных устройствах, так и на компьютерах, планшетах. При этом не для доступа в интерфейс достаточно находится в одной локальной сети с роботом и перейти по ip-адресу робота. Также, при необходимости данную систему можно конвертировать в системное приложение.

Веб-сервис будет обладать элементами управления и вывода информации:

- Кнопки;

- ползунки;
- джойстики;
- система считывания нажатий клавиатуры;
- окно вывода потокового видео с камеры;
- вывод информации о заряде батареи;
- вывод информации о текущем положении робота.

Элементы интерфейса должны реагировать на взаимодействие передавать сообщения на робота, который в свою очередь будет их обрабатывать и каким-то образом реагировать.

В нашем случае робот управляется с помощью системы ROS, которая основывается на нодах (программных единицах) и системах общения между нодами. Одной из подобных стандартизированных систем общения является Topic. Topic — это система, которая состоит из Подписчика и Издателя. Вне нод существует некоторая структура данных, которая обновляет свое значение с заданным периодом. Различные ноды могут как публиковать туда свои данные (Издатель), так и считывать их (Подписчик). Таким образом у нас будет несколько топиков, некоторые из которых заранее определены системой робота:

- /cmd_vel - направление движения робота в пространстве;
- /button - информации о состоянии батареи;
- /client_count - количество клиентов, подключенных к серверу;
- а также несколько других топиков, отвечающих за специфические функции.

Таким образом взаимодействие с элементами интерфейса должно изменять значения этих топиков соответствующим образом. Стоит упомянуть, что топик — это не единственный способ взаимодействия, но самый простой. В данном случае мы будет рассматривать только взаимодействие с топиками.

Такие элементы интерфейса, как "окно камеры" и "количество клиентов" должны наоборот брать данные из этих топиков и выводить отформатированную информации на веб-страницу.

Инструментарий

В ходе разработки концепции были описаны принципы работы интерфейса. Далее необходимо подобрать соответствующие программные инструменты для реализации разработанной концепции.

Было решено, что для разработки веб-сервиса нет необходимости использовать фреймворки и базы данных, так как сервис будет представлять собой простой набор веб-страниц с четко обозначенным количеством элементов. Сами веб-страницы будут разрабатываться на HTML (Язык гипертекстовой разметки), CSS (Каскадные таблицы стилей) и JavaScript (Язык программирования). HTML будет отвечать за расположение элементов на странице и описывать общее содержание страницы. CSS создаст возможность красивого оформления элементов. JavaScript будет отвечать за отклики элементов интерфейса и взаимодействие страницы с роботом.

Для разработки веб-сервиса нам понабиться следующий набор программных инструментов:

- VS Code - редактор кода[2];

- roslib.min.js - скрипт взаимодействия веб-сервиса и ROS [3];

- eventemitter2.min.js - скрипт обработки событий;

- keyboardteleop.min.js - скрипт обработки нажатий клавиатуры.

Со стороны робота необходимо принимать и передавать сообщения. За эту часть системы будет отвечать следующие пакеты ROS:

- usb_cam - пакет снятия данных с камеры[4];

- rosbridge_suite - пакет для связи веб-сервиса и системы ROS[5];

- web_video_server - пакет для публикации потоковых графических сообщений [6].

Итоговый разработанный сервис нужно будет опубликовать в локальную сеть и поддерживать его работу. За эту часть будет отвечать веб-сервер nginx [7].

Разработка

В качестве редактора кода, как было сказано раньше будет использоваться VS Code. Данный редактор обладает мужеством полезных для разработки плагинов, что существенно упростит разработку. Нам понадобится следующий набор плагинов:

- Live Server - для быстрой публикации веб-страницы и отслеживания изменений;

- Prettier - для удобного оформления кода;

- HTML CSS Support - для поддержки шаблонов HTML кода.

После подготовки редактора следует создать новый файл, основной файл веб-страницы. Его принято называть index.html.

В общем случае у нас будет восемь основных элементов страницы:

- header - для навагации между страницами;

- info - для описания инструкции использования сервиса;

- video - для публикации потокового видео;

- buttons - набор кнопок;

- joystick - двумерный ползунок для управления роботом;

- speed - одномерный ползунок для изменения скорости робота;

- footer - для описания состояния робота.

Условно разобьем страницы на 7-мь элементов «header» и «info» во всю ширину страниц, а остальные элементы в строчку. Первые два элемента будет в теге «header» и теге «detals», соответственно. Остальные теги объединим в единый «div» и отдельные для всех 4-х. Последний элемент «footer» будет в одноименном теге. Каждому «div» присвоим класс «inline» и класс, соответствующий названию элемента. С помощью этих классов мы будет редактировать элементы.

В блоке «buttons» создадим 6 кнопок. К каждой привяжем на нажатие вызов функции «publish(x)», где x - номер кнопки. В блоке «speed» создадим элемент ввода типа «range». В блок «video» вставим картинку без указания источника. Элементу «img» присвоим «id="video"». В элемент «joystick»

добавим еще один «div» с индексом «joystick». В «footer» вставим простой параграф и объект «span» с «id = “mess”».

Таким образом итоговый HTML код страницы будет выглядеть следующим образом (Листинг 1)

Листинг 1 – HTML код страницы index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>Control</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
    <script src='main.js'></script>
    <script
src="https://static.robotwebtools.org/roslibjs/current/roslib.min.js"></script>
    <script
src="https://static.robotwebtools.org/EventEmitter2/current/eventemitter2.min.js"><
/script>
    <script
src="https://static.robotwebtools.org/keyboardteleopjs/current/keyboardteleop.min.j
s"></script>
    <script
src="https://yoannmoinet.github.io/nipplejs/javascripts/nipplejs.js"></script>
</head>
```

Продолжение листинга 1

```
<body>
    <!-- HEADER -->
    <header>
        <h1>ROS control panel</h1>
    </header>
    <!-- INFO -->
    <details>
        <summary>Info</summary>
        Для управленяи роботом используйте клавиши WASD или стик в правой части
экрана
    </details>
    <!-- CONTROL -->
```

```
    <div class="content">
        <div class="inline buttons" >
        <button onclick="publish(1)"> Function 1 </button>
        <button onclick="publish(2)"> Function 2 </button>
        <button onclick="publish(3)"> Function 3 </button>
        <button onclick="publish(4)"> Function 4 </button>
        <button onclick="publish(5)"> Function 5 </button>
        <button onclick="publish(6)"> Function 6 </button>
        </div>
        <div class="inline video">
            <img src="" alt="" id="video" />
        </div>
        <div class="inline speed">
            <input     type="range"     min="15"     max="80"     id="robot-speed"
orient="vertical">
        </div>
        <div class="inline joystick">
            <div id="joystick"></div>
        </div>
    </div>
    <!-- FOOTER -->
    <footer>
        <p>Количество подключенных клиентов: <span id="mess"></span></p>
    </footer>
</body>
</html>
```

Далее нам необходимо разработать CSS код. Он необходим для оформления элементов страницы. С помощью него можно создать красивый и единообразный вид страницы для всех устройств или наоборот, адаптировать интерфейс для удобства использования. CSS код представлен в листинге 2.

Листинг 2 – CSS код файла main.css

```
body{
    margin-left: 5%;
    margin-right: 5%;
}
details{
    margin-top: 10px;
}
div.content{
```

Продолжение листинга 2

```
    margin-top: 10px;
    display: grid;
    grid-template-columns: 25% 50% 5% 25%;
}
div.inline {
    display: inline-block;
    height: 60%;
}
div.buttons{
    height: 70vh;
}
div.buttons button{
    margin: 5%;
    width: 90%;
    height: 10%;
}
div.video img{
    border-radius: 10px;
    width: 100%;
    height: 70vh;
    margin-top: 5px;
}
div.speed{
    margin-top: 5px;
    height: 70vh;
}
div.speed input[type="range"]{
    width: 50%;
    height: 70vh;
}
header{
    background-color: yellowgreen;
    padding-left: 1%;
    border-radius: 10px;
    font-size: 90%;
}
footer{
    margin-top: 50px;
    padding-top: 1px;
```

```
    padding-bottom:  1px;
    padding-left: 10px;
    padding-right: 10px;
    border-radius: 10px;
    background-color: brown;
}
```

Теперь необходимо запрограммировать разработанные элементы интерфейса и организовать связь с системой ROS. Начнем с установки пакетов ROS. Команды для установки и запуска пакетов приведены в листинге 3. Каждую команду следует выполнять в отдельном терминальном окне. Так как запущенные пакеты и ядра ROS будут выполнять в терминале, в котором запущена команда. Эти процессы можно перенести в фоновый режим.

Листинг 3 – команды установки и запуска пакетов ROS

```
$ sudo apt install python-tornado python-pip ros-noetic-rosbridge-suite ros- noetic
-web-video-server nginx ros-noetic-usb-cam
$ roscore
$ roslaunch rosbridge_server rosbridge_websocket.launch
$ rosrun usb_cam usb_cam_node
$ rosrun web_video_server web_video_server
```

Код JS программы можно наблюдать на листинге 4.

Листинг 4 – JS кода файла main.js

```
var twist;
var cmdVel;
var publishImmidiately = true;
var robot_IP;
var manager;
var teleop;
var ros;

function moveAction(linear, angular) {
    if (linear !== undefined && angular !== undefined) {
        twist.linear.x = linear;
        twist.angular.z = angular;
    } else {
        twist.linear.x = 0;
        twist.angular.z = 0;
    }
```

```
    cmdVel.publish(twist);
}

function publish(number) {
    if (number != undefined) {
        num.data = number
    }
    buttons.publish(num);
}



function init() {
    // Init message with zero values.
    num = new ROSLIB.Message({
        data : 0
    });

    buttons = new ROSLIB.Topic({
        ros:ros,
        name: '/buttons',
        messageType: 'std_msgs/Int16'
    });

    twist = new ROSLIB.Message({
        linear: {
            x: 0,
            y: 0,
            z: 0
        },
```

Продолжение листинга 4

```
        angular: {
            x: 0,
            y: 0,
            z: 0
        }
    });

    // Init topic object
```

```
    cmdVel = new ROSLIB.Topic({
        ros: ros,
        name: '/cmd_vel',
        messageType: 'geometry_msgs/Twist'
    });

    listener = new ROSLIB.Topic({
        ros: ros,
        name: '/client_count',
        messageType: 'std_msgs/Int32'
    });

    // Register publisher within ROS system
    buttons.advertise();
    cmdVel.advertise();
}



function sub(){
    listener.subscribe(function(message) {
        console.log('Received message on ' + listener.name + ': ' + message.data);
        document.getElementById('mess').textContent = message.data;
        listener.unsubscribe();
    });
}

setInterval(sub, 1000);



function initTeleopKeyboard() {
    // Use w, s, a, d keys to drive your robot

    // Check if keyboard controller was aready created
    if (teleop == null) {
        // Initialize the teleop.
        teleop = new KEYBOARDTELEOP.Teleop({
            ros: ros,
            topic: '/cmd_vel'
        });
    }
```

```
    // Add event listener for slider moves
    robotSpeedRange = document.getElementById("robot-speed");
    robotSpeedRange.oninput = function () {
        teleop.scale = robotSpeedRange.value / 100
    }
}

function createJoystick() {
    // Check if joystick was aready created
    if (manager == null) {
        joystickContainer = document.getElementById('joystick');
```

Продолжение листинга 4

```
// joystck configuration, if you want to adjust joystick, refer to:
        // https://yoannmoinet.github.io/nipplejs/
        var options = {
            zone: joystickContainer,
            position: { left: '87%', top: '50%' },
            mode: 'static',
            size: 150,
            color: '#0066ff',
            restJoystick: true
        };
        manager = nipplejs.create(options);
        // event listener for joystick move
        manager.on('move', function (evt, nipple) {
            // nipplejs returns direction is screen coordiantes
            // we need to rotate it, that dragging towards screen top will move robot
forward
            var direction = nipple.angle.degree - 90;
            if (direction > 180) {
                direction = -(450 - nipple.angle.degree);
            }
            // convert angles to radians and scale linear and angular speed
            // adjust if youwant robot to drvie faster or slower
            var lin = Math.cos(direction / 57.29) * nipple.distance * 0.005;
            var ang = Math.sin(direction / 57.29) * nipple.distance * 0.05;
            // nipplejs is triggering events when joystic moves each pixel
            // we need delay between consecutive messege publications to
            // prevent system from being flooded by messages
```

```
            // events triggered earlier than 50ms after last publication will be
dropped
            if (publishImmidiately) {
                publishImmidiately = false;
                moveAction(lin, ang);
                setTimeout(function () {
                    publishImmidiately = true;
                }, 50);
            }
        });
        // event litener for joystick release, always send stop message
        manager.on('end', function () {
            moveAction(0, 0);
        });
    }
}

window.onload = function () {
    // determine robot address automatically
    // robot_IP = location.hostname;
    // set robot address statically
    robot_IP = "localhost";

    // // Init handle for rosbridge_websocket
    ros = new ROSLIB.Ros({
        url: "ws://" + robot_IP + ":9090"
    });

    init();
    // get handle for video placeholder
    video = document.getElementById('video');
    // Populate video source
    video.src = "http://" + robot_IP + ":8080/stream?topic=/usb_cam/image_raw";
```

Продолжение листинга 4

```
video.onload = function () {
        // joystick and keyboard controls will be available only when video is
correctly loaded
        createJoystick();
        initTeleopKeyboard();
    };
```

```
}
```

После выполнения всех команд запуска пакетов и создания файла JS следует выгрузить наш сервис в nginx. Для этого следует отредактировать файл «/etc/nginx/sites-enable/default». В строке «root /var/www/html;» следует изменить путь к директории на директорию с разработанным сервисом. Далее перезагрузить демона командой «sudo systemctl restart nginx». После этого ваш сервис будет располагаться на вашем ip-адресе в сети и порте 80.

Итоговый вид сервиса можно наблюдать на рисунке 1 и рисунке 2.

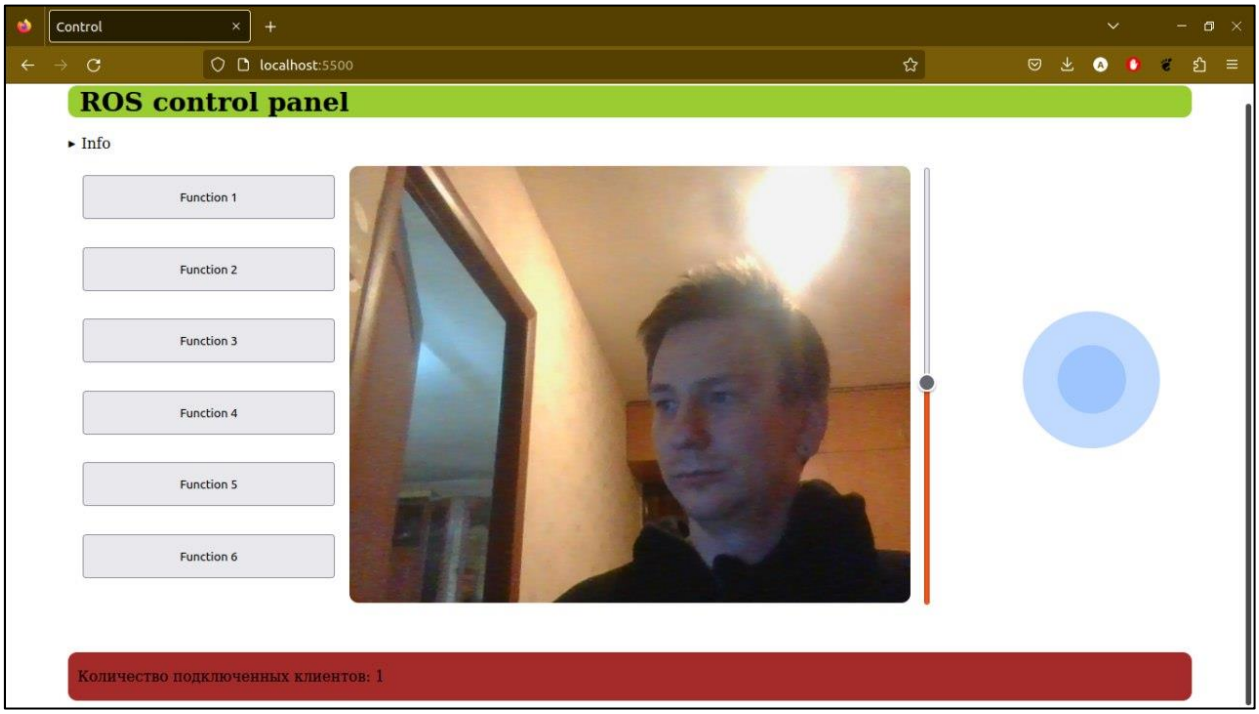

Рисунок 1 – веб-интерфейс в полноэкранном режиме

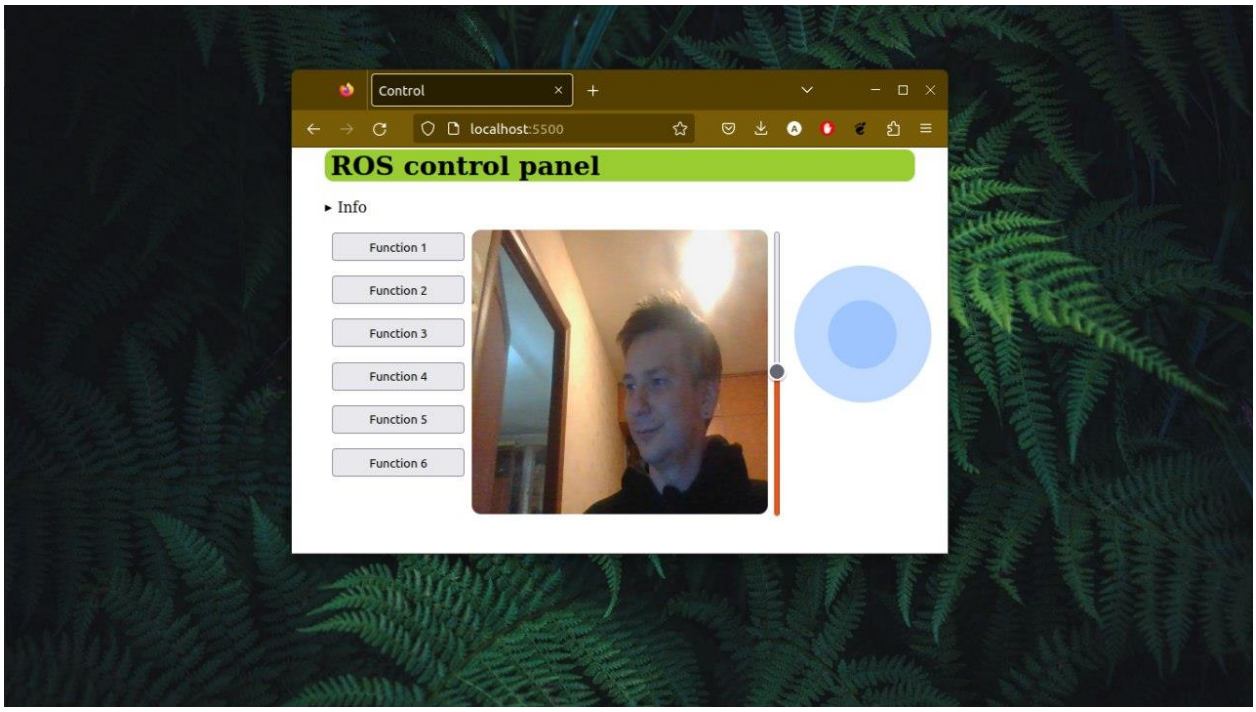

Рисунок 2 – веб-интерфейс в маленьком окне

Заключение

В результате проделанной работы была разработана система веб-интерфейса управления робота. Были выведены все шаблонные элементы интерфейса, а также разработана полная двухсторонняя связь с фреймворком ROS. В качестве доработки сервиса возможно улучшение визуально составляющей интерфейса и его адаптивности, для повышения удобства использования на всех устройствах. Также интерфейс необходимо адаптировать под ваши задачи.

Литература

1. ROS.org [Электронный ресурс]: [офиц. сайт] Documentation – Электрон. дан. – Режим доступа: http://wiki.ros.org/, свободный (дата обращения: 20.02.2023). – Загл. с экрана;
2. Visual Studio Code [Электронный ресурс]: [офиц. сайт] / Getting Started – Электрон. дан. – Режим доступа: https://code.visualstudio.com/docs, свободный (дата обращения: дата обращения: 20.02.2023). – Загл. с экрана;
3. Robot Web Tools [Электронный ресурс]: [офиц. сайт] / Libraries –

Электрон. дан. – Режим доступа: http://robotwebtools.org/, свободный (дата обращения: 30.12.2022). – Загл. с экрана;

4. ROS.org [Электронный ресурс]: [офиц. сайт] / usb_cam – Электрон. дан. – Режим доступа: http://wiki.ros.org/usb_cam, свободный (дата обращения: дата обращения: 20.02.2023). – Загл. с экрана;
5. ROS.org [Электронный ресурс]: [офиц. сайт] / rosbridge_suite – Электрон. дан. – Режим доступа: http://wiki.ros.org/rosbridge_suite, свободный (дата обращения: дата обращения: 20.02.2023). – Загл. с экрана;
6. ROS.org [Электронный ресурс]: [офиц. сайт] / web_video_server – Электрон. дан. – Режим доступа: http://wiki.ros.org/web_video_server, свободный (дата обращения: дата обращения: 20.02.2023). – Загл. с экрана;
7. NGINX [Электронный ресурс]: [офиц. сайт] / Introduction – Электрон. дан. – Режим доступа: http://nginx.org/ru/docs/, свободный (дата обращения: дата обращения: 20.02.2023). – Загл. с экрана;

Literature

1. ROS.org [Electronic resource]: [official. site] Documentation - Electronic data – Access mode: http://wiki.ros.org/, free (date of access: 02/20/2023). - From home screen;
2. Visual Studio Code [Electronic resource]: [official. site] / Getting Started - Electronic data – Access mode: https://code.visualstudio.com/docs, free (date of access: date of access: 02/20/2023). - From home screen;
3. Robot Web Tools [Electronic resource]: [official. site] / Libraries - Electronic data – Access mode: http://robotwebtools.org/, free (date of access: 12/30/2022). - From home screen;
4. ROS.org [Electronic resource]: [official. site] / usb_cam - Electronic data – Access mode: http://wiki.ros.org/usb_cam, free (date of access: date of access: 02/20/2023). - From home screen;
5. ROS.org [Electronic resource]: [official. site] / rosbridge_suite – Electronic data – Access mode: http://wiki.ros.org/rosbridge_suite, free (date of access:

date of access: 02/20/2023). - From home screen;

6. ROS.org [Electronic resource]: [official. site] / web_video_server - Electronic data – Access mode: http://wiki.ros.org/web_video_server, free (date of access: date of access: 02/20/2023). - From home screen;
7. NGINX [Electronic resource]: [official. site] / Introduction - Electronic data – Access mode: http://nginx.org/ru/docs/, free (date of access: date of access: 02/20/2023). - From home screen;

Для цитирования: *Стахеева А.А., Крайников А. Н.* РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА УПРАВЛЕНИЯ РОБОТОМ НА ОСНОВЕ ROS// Научный сетевой журнал «Столыпинский вестник» №2/2023.