Poznan University of Technology

Faculty of Control, Robotics and Electrical Engineering

Institute of Robotics and Machine Intelligence

Division Control and Industrial Electronics

Report

EL3:Networks and distributed control systems (laboratory)[WARiE_2022-23_ACR_Dz_1_6_D_LUCZAK_22/23]

(Full name of the course in the eKursy system)

Web interface jQuery and CSS

(The topic of the class)

Volodymyr Nikitin

(Author volodymyr.nikitin@student. put.poznan.pl)

Rohan Sanki

(Author rohan.sanki@student. put.poznan.pl)

form of classes:- Laboratory

Teacher:

PhD Dominik Łuczak

dominik.Luczak@put.poznan.pl

Poznań *09-05-2023* 15-10

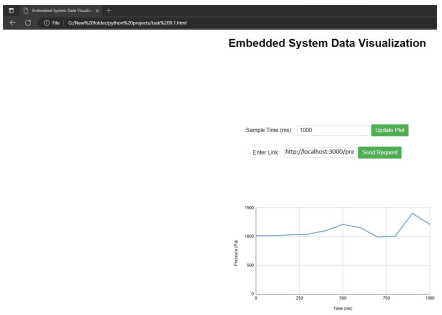
(*date and time of class*)

Contents

1. Create REST client applications for a web browser (Firefox, Chrome, etc.) for embedded system measurement data visualization

Results were presented in Table 1.

Table 1

(a) The interface should contain timeseries plots of at least two measurement quantities, e.g. angular orientation (RPY) or temperature, pressure and humidity from the Sense Hat add-on. Put on the chart all the necessary information for the correct and unambiguous interpretation of the data. Use previously prepared graphic interface prototype designs.
 	
(b) The application should cyclically request data from server and display responses in the form of timeseries plot. The sample time should be dened by the user.
 	
(c) Run and test app on a selected web browser. Make sure the network communication is working properly. You can use a physical or virtual embedded device for testing, or a server mock in the form of a local CGI script generating random synthetic measurement data.
	Here's the Html code:-
	
	<!DOCTYPE html>
	<html>
	<head>
	<title>Embedded System Data Visualization</title>
	<link rel="stylesheet" type="text/css" href="style.css">
	<style>
	body {
	font-family: Arial, sans-serif;
	margin: 0;
	padding: 0;
	}
	h1 {
	text-align: center;
	margin-top: 20px;
	margin-bottom: 30px;
	}
	#plot-container {
	width: 800px;
	margin: 0 auto;
	}
	#controls {
	text-align: center;
	margin-top: 30px;
	margin-bottom: 30px;
	}
	#link-request {
	text-align: center;
	margin-top: 30px;
	margin-bottom: 30px;
	}
	label {
	margin-right: 10px;
	}
	input[type="text"] {
	padding: 5px;
	border: 1px solid #ccc;
	border-radius: 3px;
	font-size: 16px;
	}
	button {
	padding: 8px 12px;
	border: none;
	border-radius: 3px;
	background-color: #4CAF50;
	color: #fff;
	font-size: 16px;
	cursor: pointer;
	}
	button:hover {
	background-color: #3e8e41;
		

```
    }
   </style>
  </head>
  <body>
   <h1>Embedded System Data
Visualization</h1>
   <div id="plot-container">
    <canvas id="plot"></canvas>
   </div>
   <div id="controls">
    <label for="interval">Sample Time
(ms):</label>
    <input type="text" id="interval"
value="1000">
    <button id="update">Update
Plot</button>
   </div>
   <div id="link-request">
    <label for="link">Enter Link:</label>
    <input type="text" id="link">
    <button id="send">Send
Request</button>
   </div>
   <script
src="https://cdn.plot.ly/plotly-latest.min.js">
</script>
   <script src="script.js"></script>
  </body>
</html>
```

Here's the CSS:

```
/* Set background color and font for the
whole page */
body {
  background-color: #f2f2f2;
  font-family: Arial, sans-serif;
}

/* Center the heading */
h1 {
  text-align: center;
}

/* Style the plot container */
#plot-container {
  margin: 0 auto;
  width: 80%;
  height: 400px;
}

/* Style the controls container */
#controls {
  margin: 20px auto;
  width: 80%;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Style the label and input elements */
label, input {
```

```
  font-size: 18px;
}

label {
  margin-right: 10px;
}

input {
  padding: 5px;
  border-radius: 5px;
  border: none;
}

/* Style the update button */
#update {
  padding: 10px;
  border-radius: 5px;
  border: none;
  background-color: #4CAF50;
  color: white;
  font-size: 18px;
  cursor: pointer;
}

#update:hover {
  background-color: #3e8e41;
}

/* Style the link request container */
#link-request {
  margin: 20px auto;
  width: 80%;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Style the send button */
#send {
  padding: 10px;
  border-radius: 5px;
  border: none;
  background-color: #008CBA;
  color: white;
  font-size: 18px;
  cursor: pointer;
}

#send:hover {
  background-color: #007BA7;
}
```

Here's the JavaScript Code:

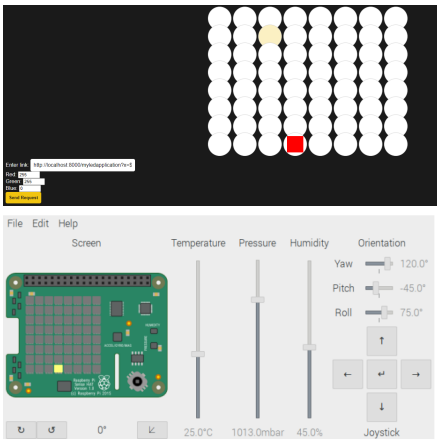
```
const chart =
document.getElementById('chart').getConte
xt('2d');
let data = [];

const config = {
  type: 'line',
  data: {
```

```
    labels: [],
    datasets: [
      {
        label: 'Measurement 1',
        data: [],
        borderColor: 'rgba(255, 99, 132, 1)',
        fill: false
      },
      {
        label: 'Measurement 2',
        data: [],
        borderColor: 'rgba(54, 162, 235, 1)',
        fill: false
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    scales: {
      xAxes: [{
        display: true,
        scaleLabel: {
          display: true,
          labelString: 'Time'
        }
      }],
      yAxes: [{
        display: true,
        scaleLabel: {
          display: true,
          labelString: 'Value'
        }
      }]
    }
  }
};

const myChart = new Chart(chart, config);

function requestData() {
  const request = new XMLHttpRequest();
  request.open('GET',
'http://localhost:8000/getdata', true);
  request.onload = function() {
    if (this.status >= 200 && this.status <
400) {
      const response =
JSON.parse(this.response);
      data.push(response);
      if (data.length > 10) {
        data.shift();
      }
      myChart.data.datasets[0].data =
data.map((d) => d.measurement1);
      myChart.data.datasets[1].data =
data.map((d) => d.measurement2);
      myChart.data.labels = data.map((d) =>
d.time);
      myChart.update();
    }
  };
  request.send();
}

setInterval(requestData, 1000);
```

2. Expand client application with activity to control user output device (e.g. LED display).

Results were presented in Table 2.

Table 2

(a) The interface should allow setting the state of all available user outputs (e.g. the LED matrix from the Sense Hat add-on). The interface should clearly communicate whether the current user settings correspond to the output states (i.e. whether the user has applied the last changes). Use previously prepared graphic interface prototype designs.
 	
(b) The application should send a request to the server containing control commands. Use an adequate HTTP method.
 	
(c) Run and test app on a physical or virtual mobile device. Make sure the network communication is working properly. You can use a physical or virtual embedded device for testing, or a server mock in the form of a local CGI script saving control commands to text le
	Here's the Html code:-
	
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>LED Control</title>`
	`<style>`
	`.grid-container { display: grid; grid-template-columns: repeat(8, 1fr); grid-gap: 5px; margin: 20px auto; max-width: 500px; text-align: center; }`
	`.grid-item { background-color: #fff; border: 1px solid #ccc; padding: 10px; font-size: 24px; }`
	`.grid-item input[type="checkbox"] { display: none; }`
	`.grid-item input[type="checkbox"]:checked + label { background-color: #f00; }`
	`.grid-item label { display: block; width: 100%; height: 100%; cursor: pointer; }`
	`</style>`
	`</head>`
	`<body>`
	`<div class="grid-container">`
	`<div class="grid-item"><input type="checkbox" id="led-0"><label`
		

```
for="led-0"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-1"><label
for="led-1"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-2"><label
for="led-2"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-3"><label
for="led-3"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-4"><label
for="led-4"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-5"><label
for="led-5"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-6"><label
for="led-6"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-7"><label
for="led-7"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-8"><label
for="led-8"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-9"><label
for="led-9"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-10"><label
for="led-10"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-11"><label
for="led-11"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-12"><label
for="led-12"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-13"><label
for="led-13"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-14"><label
for="led-14"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-15"><label
for="led-15"></label></div>
                <div
class="grid-item"><input type="checkbox"
id="led-16"><label
for="led-16"></label></div>
                <div
class="grid-item"><input type="checkbox"
```

```
id="led-17"><label
for="led-17"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-18"><label
for="led-18"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-19"><label
for="led-19"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-20"><label
for="led-20"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-21"><label
for="led-21"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-22"><label
for="led-22"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-23"><label
for="led-23"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-24"><label
for="led-24"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-25"><label
for="led-25"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-26"><label
for="led-26"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-27"><label
for="led-27"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-28"><label
for="led-28"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-29"><label
for="led-29"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-30"><label
for="led-30"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-31"><label
for="led-31"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-32"><label
for="led-32"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-33"><label
for="led-33"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-34"><label
for="led-34"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-35"><label
for="led-35"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-36"><label
for="led-36"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-37"><label
for="led-37"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-38"><label
for="led-38"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-39"><label
```

```
for="led-39"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-40"><label
for="led-40"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-41"><label
for="led-41"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-42"><label
for="led-42"></label></div>
<div class="grid-item"><input
type="checkbox" id="led-43"><label
for="led
<div class="grid-item"><input
type="checkbox" id="led-44"><label
for="led-44"></label></div>
   <div class="grid-item"><input
type="checkbox" id="led-45"><label
for="led-45"></label></div>
   <div class="grid-item"><input
type="checkbox" id="led-46"><label
for="led-46"></label></div>
   <div class="grid-item"><input
type="checkbox" id="led-47"><label
for="led-47"></label></div>
   <div class="grid-item"><input
type="checkbox" id="led-48"><label
for="led-48"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-49"><label
for="led-49"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-50"><label
for="led-50"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-51"><label
for="led-52"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-53"><label
for="led-53"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-54"><label
for="led-54"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-55"><label
for="led-55"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-56"><label
for="led-56"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-57"><label
for="led-57"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-58"><label
for="led-58"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-59"><label
for="led-59"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-60"><label
for="led-60"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-61"><label
for="led-61"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-62"><label
for="led-62"></label></div>
```

```
  <div class="grid-item"><input
type="checkbox" id="led-63"><label
for="led-63"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-64"><label
for="led-64"></label></div>
  <div class="grid-item"><input
type="checkbox" id="led-65"><label
for="led-65"></label></div>
  </div>
  <div class="input-section">
  <label for="link-input">Enter link:</label>
  <input type="text" id="link-input">
  <br>
  <label for="red-input">Red:</label>
  <input type="number" id="red-input"
min="0" max="255" value="0">
  <br>
  <label for="green-input">Green:</label>
  <input type="number" id="green-input"
min="0" max="255" value="0">
  <br>
  <label for="blue-input">Blue:</label>
  <input type="number" id="blue-input"
min="0" max="255" value="0">
  <br>
  <button onclick="sendRequest()">Send
Request</button>
</div>
```

Here's the CSS code:

```
body {
  background-color: #1a1a1a;
  color: #fff;
  font-family: Arial, sans-serif;
}

h1 {
  font-size: 2em;
  font-weight: bold;
  color: #f1c40f;
}

h2 {
  font-size: 1.5em;
  color: #f1c40f;
}

p {
  font-size: 1em;
  line-height: 1.5em;
}

button {
  background-color: #f1c40f;
  color: #1a1a1a;
  font-weight: bold;
```

```
  border: none;
  border-radius: 5px;
  padding: 10px;
  cursor: pointer;
}

button:hover {
  background-color: #fff;
  color: #1a1a1a;
}

.grid-container {
  display: grid;
  grid-template-columns: repeat(8, 50px);
  grid-template-rows: repeat(8, 50px);
  grid-gap: 5px;
  justify-content: center;
  align-items: center;
  margin-top: 20px;
}

.grid-item {
  background-color: #444;
  border-radius: 50%;
  width: 50px;
  height: 50px;
  transition: background-color 0.3s
ease-in-out;
}

.grid-item:hover {
  background-color: #f1c40f;
}

#link-input {
  width: 300px;
  padding: 10px;
  border: none;
  border-radius: 5px;
  margin-right: 10px;
  font-size: 1em;
  color: #1a1a1a;
  background-color: #fff;
}

.color-selector {
  display: flex;
  align-items: center;
  margin-top: 20px;
}

.color-selector label {
  font-size: 1.2em;
  margin-right: 10px;
  color: #f1c40f;
}

.color-selector input[type="number"] {
  width: 50px;
  margin-right: 10px;
  font-size: 1.2em;
  padding: 5px;
  border: none;
  border-radius: 5px;
  background-color: #444;
  color: #fff;
}
```

```
.color-selector
input[type="number"]::-webkit-inner-spin-b
utton,
.color-selector
input[type="number"]::-webkit-outer-spin-b
utton {
  -webkit-appearance: none;
  margin: 0;
}

.color-selector input[type="number"]:focus
{
  outline: none;
  box-shadow: 0 0 3px 3px #f1c40f;
}
```

Here's the JavaScript Code:

```
function sendRequest() {
  var link =
document.getElementById("link-input").val
ue;
  var red =
document.getElementById("red-input").valu
e;
  var green =
document.getElementById("green-input").v
alue;
  var blue =
document.getElementById("blue-input").val
ue;

  var checked_leds = [];
  var checkboxes =
document.querySelectorAll('input[type=che
ckbox]:checked');
  for (var i = 0; i < checkboxes.length; i++) {

checked_leds.push(parseInt(checkboxes[i].i
d.split("-")[1]));
  }

  var data = {
    link: link,
    red: red,
    green: green,
    blue: blue,
    checked_leds: checked_leds
  };

  // Send request to server with data using an
AJAX request
  // ...
}
```

3. Expand your application with a module (web page) to configure and save user settings.

Results were presented in Table 3.

Table 3

<table>
<tr>
<td>(a) The configuration page should contain basic application settings, such as port number, server API version (convenient for the developer), requests sample time, maximum number of saved samples, etc.

(b) Configuration information should be saved on server as a JSON text le.

(c) After reloading web page, saved user's settings should be read from the server.</td>
<td><u>Here’s html code:</u>

<!DOCTYPE html>
<html>
<head>
<title>Configuration Page</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<h1>Configuration Page</h1>
<form>
<fieldset>
<legend>Server Settings</legend>
<label for="port">Port Number:</label>
<input type="number" id="port" name="port" min="1" max="65535" value="8000">

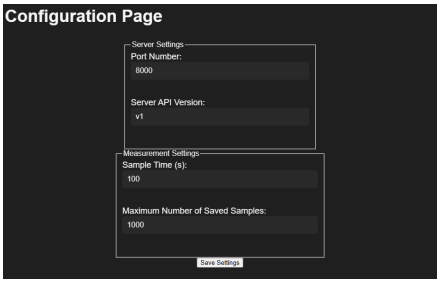
<label for="api_version">Server API Version:</label>
<input type="text" id="api_version" name="api_version" value="v1">

</fieldset>
<fieldset>
<legend>Measurement Settings</legend>
<label for="sample_time">Sample Time (s):</label>
<input type="number" id="sample_time" name="sample_time" min="1" max="3600" value="100">

<label for="max_samples">Maximum Number of Saved Samples:</label>
<input type="number" id="max_samples" name="max_samples" min="10" max="10000" value="1000">

</fieldset>
<button type="submit">Save Settings</button>
</form>
<script src="config.js"></script>
</body>
</html>

<u>Here’s CSS code:-</u>

body {
background-color: #1f1f1f;</td>
<td> 

▼ object {4}
portNumber : 8000
apiVersion : v1
sampleTime : 100
maxSamples : 1000</td>
</tr>
</table>

```
  color: #f5f5f5;
  font-family: Arial, sans-serif;
}

.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 50px;
}

h1 {
  font-size: 36px;
  margin-bottom: 30px;
}

form {
  display: flex;
  flex-direction: column;
  align-items: center;
}

label {
  font-size: 18px;
  margin-bottom: 10px;
}

input[type="text"], input[type="number"],
select {
  width: 100%;
  padding: 10px;
  margin-bottom: 20px;
  border-radius: 5px;
  border: none;
  background-color: #292929;
  color: #f5f5f5;
  font-size: 16px;
}

input[type="submit"] {
  padding: 10px 20px;
  border-radius: 5px;
  border: none;
  background-color: #3cb371;
  color: #f5f5f5;
  font-size: 18px;
  cursor: pointer;
}

input[type="submit"]:hover {
  background-color: #2e8b57;
}

.alert {
  margin-top: 20px;
  padding: 10px;
  background-color: #d9534f;
  color: #f5f5f5;
  border-radius: 5px;
  text-align: center;
  display: none;
}

.success {
  background-color: #5cb85c;
  border-color: #4cae4c;
  color: #f5f5f5;
}
```

```
.error {
  background-color: #d9534f;
  border-color: #d43f3a;
  color: #f5f5f5;
}
```

Here's the JavaScript Code:

```
const form =
document.querySelector('form');
const portInput =
document.querySelector('#port');
const apiVersionInput =
document.querySelector('#api_version');
const sampleTimeInput =
document.querySelector('#sample_time');
const maxSamplesInput =
document.querySelector('#max_samples');

// Get the current settings from the server
fetch('/settings.json')
   .then(response => response.json())
   .then(settings => {
      portInput.value = settings.port;
      apiVersionInput.value =
settings.api_version;
      sampleTimeInput.value =
settings.sample_time;
      maxSamplesInput.value =
settings.max_samples;
   })
   .catch(error => console.error(error));

// Save the settings to the server
form.addEventListener('submit', event => {
   event.preventDefault();
   const settings = {
      port: portInput.value,
      api_version: apiVersionInput.value,
      sample_time: sampleTimeInput.value,
      max_samples: maxSamplesInput.value
   };
   fetch('/settings', {
      method: 'PUT',
      headers: {
         'Content-Type': 'application/json'
      },
      body: JSON.stringify(settings)
   })
      .then(response => console.log('Settings
saved successfully.'))
      .catch(error => console.error(error));
});
```

Literature

1. Android Studio [shared on 11.05.2023]. Downloaded: https://developer.android.com/studio

2. https://ekursy.put.poznan.pl/pluginfile.php/2425341/mod_folder/content/0/AIR_ENG_L09.pdf?forcedownload=1

3.https://ekursy.put.poznan.pl/pluginfile.php/2394778/mod_folder/content/0/W08_AMiWIP%20Web%20-%20CSS%2C%20Bootsrap.pdf?forcedownload=1