Experimeht with JavaScript on Client and Server Side of the Virtual Laboratory and Visualized in Mixed Reality Using Microsoft HoloLens

Erich Stark, Erik Kučera, Pavol Bisták, Oto Haffner

DOI: 10.14313/JAMRIS_1-2018/2

Abstract:
The paper demonstrates remote control of test experiment in the virtual laboratory. This is a common problem, but another way can always be used to solve it. The paper compares several existing virtual laboratories and their possible issues at present. To develop such a new solution JavaScript technology was used on both client and server side using Node.js runtime. The modern approach is a visualization of received data in mixed reality using Microsoft HoloLens or another compatible device with Windows Mixed Reality platform.

Keywords: virtual experiment, virtual laboratory, javascript, Unity engine, virtual reality, mixed reality

1. Introduction
Practical exercises in the laboratory are an important part of the process of training people with technical background in general. Ancient Chinese philosopher Confucius once said: “Tell me, and I will forget. Show me, and I may remember. Involve me, and I will understand” [20]. We know from experience that man can learn in the fastest way when he tries things several times, and after that, he understands how it works. Unfortunately, you cannot always provide direct access to real devices to perform the experiment for researchers or students. There may be several issues: the higher price of laboratory equipment, workplace safety (depending on the experiment), or lack of qualified assistants.

In recent years, the development of virtual machines has increased mainly due to the technological evolution of software engineering. The progress of modern technology gives us the better approach to solve new challenges while creating whether the virtual systems for online teaching or specific virtual laboratories where physical processes can be simulated. In experiments conducted in a virtual environment, it is possible to share resources of this environment for more connected users who want to perform the same experiment, which would not be possible on our computers. This makes virtual laboratory a good complement to study whether research, where you can try different variations of the experiment without risk to health or destruction of the device. Later, experiments can be tested on real devices, if necessary.

2. Virtual Laboratories
At the time when the Internet was not yet widespread in use, the experiments were done in real laboratories. It was important to keep on with different safety regulations to the possibility of personal injury or damage to equipment.

Distance and lack of financial resources make real experiments difficult to perform, especially in cases where it is necessary to have some advanced and sophisticated tools. Another encountered problem is the lack of good teachers. Although at present there are already online courses that provide instructional videos, but it solves the problem only partially. Thanks to internet experiments can be structured for visualization and control remotely. Nowadays, a lot of equipment already provides an interface to connect computer and process data from it. Experimenting over the internet allows the use of resources, knowledge, software and data when physical experiments cannot [15].

In this paper, we discuss the creation of virtual laboratory (VL). Before we describe the list of technologies to create VL, we must explain what we consider under VL. Generally, we can say that VL is a computer program, where students interact with the experiment by the computer via the Internet as it is depicted in the Fig. 1.

![Fig. 1. The difference between a face-to-face and remotely controlled experiments](image)

A typical example is the simulation experiment, where the student interacts with the web/app interface. Another possibility is a remote-controlled experiment where the student interacts with the real device via the computer interface, although he can be far away. This is the case when a virtual laboratory turns into a remote laboratory.

When the web excludes the second option, so we
have the following definition: “We call it a virtual laboratory where the student interacts with the experiment, which is physically at distant from him or her and not to demand any physical reality”.

After explaining what is VL, look at the benefits they can bring. They are described in the Table 1.

**Tab. 1. Comparison of Real, Virtual and Remote Laboratories [7]**

<table>
<thead>
<tr>
<th>Laboratory Type</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>real data, interaction with real experiment, collaborative work, interaction with supervisor</td>
<td>time and place restrictions, requires scheduling, expensive, supervision required</td>
</tr>
<tr>
<td>Virtual</td>
<td>good for concept explanation, no time and place restrictions, interactive medium, low cost</td>
<td>idealized data, lack of collaboration, no interaction with real equipment</td>
</tr>
<tr>
<td>Remote</td>
<td>interaction with real equipment, calibration, realistic data, no time and place restrictions, medium cost</td>
<td>only “virtual presence” in the lab</td>
</tr>
</tbody>
</table>

People often think that the main benefit of a virtual laboratory is to replace the real one. But it is not. You cannot replace the experience of the real work with the VL. Although VL is better than no experience. VL should not be seen as providing the maximum possible interaction experience.

**2.1. Existing Solutions**

There are currently many different virtual and remote laboratories, which are used by foreign universities for teaching or research. This paper briefly reviews often used laboratories that are accessible over the Internet. A comparison of functionality and the use of technology can be seen in the Table 2, where different virtual laboratories created in the world are summarized.

There are also some examples from Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava in the Table 3.

**Tab. 2. Comparison of Virtual Laboratories Created Outside of FEI STU [12]**

<table>
<thead>
<tr>
<th>Name of VL</th>
<th>Client technology</th>
<th>Server technology</th>
<th>Simulation software</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebLab-DEUSTO</td>
<td>AJAX, Flash, Java applets, LabVIEW, Remote panel</td>
<td>Web services, Python, LabVIEW, Java, .NET, C, C++</td>
<td>Xilinx-VHDL, LabVIEW</td>
</tr>
<tr>
<td>NCSLab</td>
<td>AJAX, Flash</td>
<td>PHP</td>
<td>Matlab, Simulink</td>
</tr>
<tr>
<td>ACT</td>
<td>HTML, Java applets</td>
<td>PHP</td>
<td>Matlab, Simulink</td>
</tr>
<tr>
<td>LabShare Sahara</td>
<td>AJAX, Java applets</td>
<td>Web services, Java</td>
<td>Java</td>
</tr>
<tr>
<td>iLab</td>
<td>HTML, Active X, Java applets</td>
<td>Webservices, .NET</td>
<td>LabVIEW</td>
</tr>
<tr>
<td>RECOLAB</td>
<td>HTML</td>
<td>PHP</td>
<td>Matlab, Simulink</td>
</tr>
<tr>
<td>SLD</td>
<td>AJAX, HTML</td>
<td>Web services, PHP</td>
<td>Matlab, Simulink</td>
</tr>
</tbody>
</table>

2.2. Disadvantaged of Existing Solutions

At the beginning of the design of a virtual laboratory, it was appropriate to examine the possibilities of existing solutions. Avoiding various design issues is important. Alternatively, technologies that have been used are already outdated. Nowadays, the development of new technologies is incredibly fast. Such an analysis of existing solutions we have done in the previous section. Our aim was to create a cross-platform solution using one programming language on client and server side, which cannot be done with WCF or COM technology as in the previous solutions. JMI is only suitable for solutions where Java platform is used. The server cannot also be used with LabVIEW technology or .NET (multi-platform version – .NET core is already under development). Client solutions such as Flash, ActiveX and Java applets are no longer supported in browsers, so their use is not appropriate.

**2.3. Components of Virtual Laboratory**

There are plenty of existing laboratories, but usually, it is not possible to guarantee compatibility between them, because there is not a solid standard. Anyway, it is always possible to identify the basic components that virtual laboratories can use. Some of them can be even used more times.

Components:
- The experiment itself
- The device with possibility to control and acquiring data
- Laboratory server, which provides control, monitoring and data processing of the experiment
- Server providing connection between remote users and laboratory server, usually via the internet
- Web camera connected to a server, which can be used for remote user as a visual and audible feedback on the actual status of the experiment
- Tools enabling multi-user audio, video and chat communication
- Client software controlling and representing data of the experiment [5]

It is important to realize which of these components could be used, because for a creation of a virtual laboratory it is not necessary to have them all. Alternatively, others that are perfectly suited for a role can also be used. Sometimes it is used e.g. database server if experiments will be stored and processed later. It is also important to realize what type of VL we want to create. Certainly, differences will be in the design of single-user as opposed to multi-user VL, even with multiple experiments simultaneously. It should be considered as properly solving the scalability, potential safety issues, multi-user access and other possible issues.

3. Architecture Proposal

As the main component, Node.js was selected. It is the server which handles communication between components of VL. The parts of architecture will be explained based on Fig. 2. The data are fetched periodically from Simulink into Matlab workspace. In the beginning, it was not sure whether it would be possible to achieve to run multiplatform soft real-time Simulink based simulations. Because only Windows based solution was found directly from MathWorks. For our solution, Real-Time Pacer [19] was used that allows us to run simulations in soft real-time even under MacOS or Linux. It is used to slow down the simulation to the soft real-time.

To communicate with RESTful web service Matlab R2015a uses the built-in rarely used function webread and webwrite [14]. Firstly, the simulation must be run through the web browser, after that data will be transferred over socket.js library channel. These data will be shown in the graph of the web browser, and it is possible to save them to MongoDB database for later processing (Fig. 2).

3.1. Reference Simulation Model

For a development purpose, we used the simulation of the dynamic system called projectile motion implemented in Simulink that runs through the web interface. This simulation needs to be run with two files. The purpose of the first is the initialization of variables needed to calculate the coordinates of the point. This experiment has three parameters. The first and second parameter are initial values for simulation. The last parameter userFromWeb is not necessary for simulation itself, but it is important to identify the user who runs the simulation. This makes it possible to assign the simulation results in later processing from the database.

3.2. Experiment Handler

The second Matlab file is a handler code sending the data to Node.js. Because of its length of implementation, it is not possible to display the whole source code, so we describe only the key part.

During initialization, the URL path is set for Express.js REST API where Matlab will send the data.

The model is preloaded using the Matlab function load_system('projectile_motion'). This function searches in the current folder for projectile_motion.mdl file and sets it as the top-level model. After this initial settings, simulation must be run using the command set_param(model, 'SimulationCommand', 'Start').

In the next block of the Matlab code, it is running an infinite while loop that makes possible to collect data from the simulation to the state until it is complete. Inside of the while loop the function set_param(model, 'SimulationCommand', 'WriteDataLogs') is called, which is looking for the current top-level simulation. In the soft real-time the calculated data are written to the Matlab workspace. Without that function, data would be written only after the simulation ends.
Meanwhile, it is necessary to prepare required format of data for the web service. Thus, before sending them to the REST API, it is suitable to wrap data to the JSON structure. We used the Matlab library JSON-lab v1.2 [4].

A sequence of these two commands is required to create the desired JSON format and send it to Express.js API. Create JSON with the command

```
JSON = savejson('result', struct('user', userFromWeb, 'status', 'Running', 'data', struct('time', timeFinal, 'you', vyFinal, 'y', yFinal, 'x', xfinal)));
```

and transfer it to the service with

```
response = webwrite(URL, JSON, options)
```

The command `get_param(model, 'SimulationStatus')` is used to check current status of the simulation. If the simulation is still running the status is “running”. As soon as the status is “stopped”, the loop needs to be stopped using the `break` keyword and we know that all data is transferred to Node.js.

### 3.3. Communication Between Components

One of the aspects of the individual components of the laboratory is communication. Although in each component communication works differently, it is still based on the HTTP protocol.

The sequence diagram in the Fig. 6 shows that communication starts from the web browser. The user inserts the parameters of simulation, which are sent to StarkLab via the REST web service. This service starts Matlab on the current operating system with the necessary files and simulation parameters. Meanwhile, the user waits until Matlab starts in the background. Simulation is immediately initialized and starts sending data to StarkLab, which sends them directly to the web client from where the simulation has been originated. All the received data will be reflected in the chart, animation, and table in the web browser. This sequence is repeated until the condition contains `SimulationStatus == "running"`. After stopping the simulation, the client sends a request to save data through StarkLab directly into the document database MongoDB.

### 3.4. Run Matlab from Command Line

In the beginning, it was not clear how to run the simulation. It was necessary to determine whether Node.js allows to carry out the commands of the operating system, respectively run programs. The simulation was working in such a way that the Matlab was opened manually and we put there all the necessary initialization files, then the simulation itself. But this solution is not sufficient in terms of automation and autonomy.

It has been found that Node.js can launch any software that can run through the terminal. To simplify this workflow the `shell.js` library [2] was used which provides such an option.

The sample of code in the Fig. 3 shows how Matlab is started via Node.js route `http://localhost/matlab/run`. This route is called immediately after the form was sent with initial parameters of the experiment from the web browser.

![Fig. 3. Start Matlab in command line using shell.js library](image)

### 4. Remote Control of Experiment

#### 4.1. Web Client Created with Angular Framework

Client application was created with the JavaScript framework Angular [6] (version 1.5.5). The role of the web client was to verify the functionality of the server that sends simulated data. The functionality has been verified, and screens will be described specifically.

![Fig. 4. Login to web application](image)
is to obtain canvas context and create the object with initial data.

The plotted data at the bottom of the picture is identical to the data in the graph. The difference is in the way of implementation as animation. This animation was created using HTML Canvas technology.

The last section, where the data can be seen is a table where data were added over time as well as chart and animation before. In this table Angular databinding [6] is used to set received object as one row with their properties. As simulation runs, the Angular adds new rows to table dynamically.

This system is not only about the real-time rendering of data, but also for later viewing and processing of them. On the site of simulations, we can see all the entries for the currently logged in user - Fig. 14. The list is obtained from MongoDB using Angular $http.get(url, callback) function from web client to our Node.js server, which can have access to database.

When the one of the results is opened, the output looks the same as in Fig. 7, but it is possible to set data sampling and time of simulation. The second option is about time rendering. There are two options: to see data output immediately or soft real-time as it was firstly run.

### 5. Visualisation of Virtual Laboratory in Mixed Reality

Modern forms of education are now realized on the basis of the development of new ICT technologies (e.g. interactive applications made in 3D engine [17], virtual reality or mixed reality). Visualisation of process modelling, identification and control of complex mechatronic systems, elements and drives using virtual and mixed reality allows students to get a much better and quicker understanding of the studied subject compared to conventional teaching methods.
is to obtain canvas context and create the object with initial data. The plotted data at the bottom of the picture is identical to the data in the graph. The difference is in the way of implementation as animation. This animation was created using HTML Canvas technology.

The last section, where the data can be seen isatable where data were added over time as well as chart and animation before. In this table Angular data binding [6] is used to set received objects on one row with their properties. As a simulation runs, the Angular adds new rows to the table dynamically.

This system is not only about the real-time rendering of data, but also for later viewing and processing of them. On the site of simulations, we can see all the entries for the currently logged in user – Fig. 14. The list is obtained from MongoDB using Angular $http.get(url, callback) function from web client to our Node.js server, which can have access to the database.

When the one of the results is opened, the output looks the same as in Fig. 7, but it is possible to set data sampling and time of simulation. The second option is about timerendering. There are two options: to see data output immediately or soft real-time as it was instantly run.

Modern forms of education are now realized on the basis of the development of new ICT technologies (e.g. interactive applications made in 3D engine [17], virtual reality or mixed reality). Visualisation of process modeling, identification and control of complex mechatronic systems, elements and drives using virtual and mixed reality allows students to get a much better and quicker understanding of the studied subject compared to conventional teaching methods.

5.1. Differences Between Virtual, Augmented and Mixed Reality

Virtual reality (VR) replicates an environment that simulates a physical presence in places in the real world or an imagined world, allowing the user to interact in that world. Devices for virtual reality are Google Cardboard, HTC Vive, Oculus Rift, etc.

Augmented reality (AR) is a live, direct or in-
Mixed reality (MR) is the merging of real and virtual worlds to produce new environments and visualisations where physical and digital objects co-exist and interact in real time. MR is an overlay of synthetic content on the real world that is anchored to and interacts with the real world. The key characteristic of MR is that the synthetic content and the real-world content are able to react to each other in real time. Technologies for mixed reality are Microsoft HoloLens (Windows Mixed Reality platform), Android ARCore and Apple ARKit.

Nowadays, there is a trend of using interactive 3D applications and virtual reality in many prestigious universities.

Very interesting project is a virtual clinic [9]. This project is supported by the University of Miami or Charles R. Drew University of Medicine and Science in Los Angeles. This interactive application offers an insight into the actual functioning of a larger clinic, and they can also try to diagnose patients. Students are thus trained through a real experience with the health system, but this complex system is modelled and simulated in virtual reality.

There are also interactive applications from Ani-mech Technologies. This company offers many education modules like Virtual Car, Virtual Truck or Virtual Gearbox [16]. Using these applications students can understand the functioning of mentioned devices and they can look into their interior and detach their individual components in detail.

An absolute novelty is Microsoft HoloLens [11], the arrival of which has led to the emergence of a completely new segment of mixed reality. Mixed reality has unquestionable advantages over virtual reality, as the user perceives a real world and also a virtual world in the same time. The use of this feature is in practice undisputed and it is assumed that mixed reality will become a new standard in many areas such as education, marketing, modeling of complex mechatronic systems, etc.

For Microsoft HoloLens there are more education applications.

Application HoloTour [3] provides 360-degree spatial video of historical places like Rome or Peru. The application complements 3D models of important landmarks that have not been retained or supplementary holographic information about elements in the scene.

Application HoloAnatomy [10] allows interactive education of anatomy of the human body. The advantage is that if the application is used by more users in the same time, everyone sees the same model of a part of the human body. This allows an interaction between students that results in a significant multiplier education effect.

From technical fields there is an application called HoloEngine [1]. This application allows understanding of the complex 3D mechanical structures of the combustion engine. The application allows you to see the engine in the air, start it and even look inside it and closely monitor the mutual interaction of the mechanical parts.

5.2. Virtual Laboratory in Mixed Reality

For development Unity engine was used. Proposed Unity application for Microsoft HoloLens brings a visualisation of results from described Node.js virtual
laboratory in mixed reality. By this application, students get better insight into the results of the experiment.

It was needed to connect Unity application with Node.js laboratory. There is a free library for Unity called Socket.IO for Unity [8] which was used in the proposed application.

In Fig. 16, it is possible to see the results from the virtual laboratory in Unity engine. The application was deployed on Microsoft HoloLens. The results in mixed reality, you can see in Fig. 17.

Fig. 16. Results from virtual laboratory in Unity engine

Fig. 17. Results from virtual laboratory in mixed reality (Microsoft HoloLens)

6. Conclusion

After the experience with this kind of development, we assess that the creation of virtual laboratory platform on Node.js development was easier thanks to the use of JavaScript on the server and client side. We thought that due to the single thread loop of Node.js would handle more clients and simulations than the similar solution on a different platform. The problem was not in many of registered users, but only when we run multiple simulations in Matlab. In our test computer – MacBook Pro there was already a problem with two parallel simulations. It can be improved using a powerful server for Matlab calculations.

The work is not over yet and Starklab can be extended with another interesting functionality such as the creation of a unified protocol for data interchange. Suitable would be also interfaces for other calculation and simulation software. Matlab deployment on a separate server with an available domain would help to availability. Another interesting functionality would be uploading simulation and calculation scripts through a web interface.

The future work will also focus on additional development for Windows Mixed Reality platform and Microsoft HoloLens headset.

The source code of the virtual laboratory is available as open source at https://github.com/erichstark/.

ACKNOWLEDGEMENTS

This work has been supported by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, KEGA 030STU-4/2017, by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under the grant VEGA 1/0733/16 and VEGA 1/0819/17, and by the Tatra banka Foundation within the grant programme Quality of Education, project No. 2017vs022 (Collaborative education applications in mixed reality for mechatronics).

AUTHORS

Erich Stark* – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: erich.stark@stuba.sk, www: www.uamt.fei.stuba.sk.

Erik Kučera* – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: erik.kucera@stuba.sk, www: www.uamt.fei.stuba.sk.

Pavol Bisták – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: pavol.bistak@stuba.sk, www: www.uamt.fei.stuba.sk.

Oto Haffner – Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, Slovakia, e-mail: oto.haffner@stuba.sk, www: www.uamt.fei.stuba.sk.

*Corresponding author

REFERENCES