Communication papers of the Federated Conference on Computer Science and Information Systems, pp. 315–322
DOI: 10.15439/2017F313
ISSN 2300-5963 ACSIS, Vol. 13

Virtual Laboratory Based on Node.js Technology and Visualized in Mixed Reality Using Microsoft HoloLens

Erich Stark, Pavol Bisták, Erik Kučera, Oto Haffner and Štefan Kozák
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
Email: erich.stark@stuba.sk, erik.kucera@stuba.sk

***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.**

I. 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" [1]. 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.

II. 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.

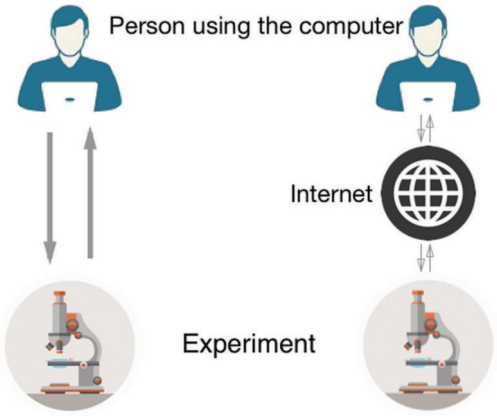

Fig. 1. The difference between a face-to-face and remotely controlled experiments

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 [2].

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.

A typical example is the simulation experiment, where the student interacts with the web/app interface. Another

TABLE I
COMPARISON OF REAL, VIRTUAL AND REMOTE LABORATORIES [3]

Laboratory Type	Advantages	Disadvantages
Real	real data, interaction with real experiment, collaborative work, interaction with supervisor	time and place restrictions, requires scheduling, expensive, supervision required
Virtual	good for concept explanation, no time and place restrictions, interactive medium, low cost	idealized data, lack of collaboration, no interaction with real equipment
Remote	interaction with real equipment, calibration, realistic data, no time and place restrictions, medium cost	only "virtual presence" in the lab

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 I.

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.

A. 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 II, where different virtual laboratories created in the world are summarized.

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

B. 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

TABLE II
COMPARISON OF VIRTUAL LABORATORIES CREATED OUTSIDE OF FEI STU [4]

Name of VL	Client technology	Server technology	Simulation software
Weblab-DEUSTO	AJAX, Flash, Java applets, LabVIEW, Remote panel	Web services, Python, LabVIEW, Java, .NET, C, C++	Xilinx-VHDL, LabVIEW
NCSLab	AJAX, Flash	PHP	Matlab, Simulink
ACT	HTML, Java applets	PHP	Matlab, Simulink
LabShare Sahara	AJAX, Java applets	Web services, Java	Java
iLab	HTML, Active X, Java applets	Webservices, .NET	LabVIEW
RECOLAB	HTML	PHP	Matlab, Simulink
SLD	AJAX, HTML	Web services, PHP	Matlab, Simulink

TABLE III
COMPARISON OF VIRTUAL LABORATORIES CREATED AT FEI STU [5]

Year	Author	Simulation software	Data flow	Client technology	Server technology
2011	R. Farkas	Matlab, Simulink, Real device	JMI Sockets	Java	Java
2012	T. Borka	Matlab, Simulink, Real device	WCF	.NET, WPF	.NET
2014	M. Kundrat	Matlab, Simulink	JMI, SOAP	HTML, JS	Tomcat, Java JSF, EJB3, MySQL
2014	T. Cerveny	Matlab, Simulink	JMI, HTTP	HTML, JS	Jetty, Java
2015	S. Varga	Matlab, Simulink	COM, HTTP	HTML, JS	.NET, PHP

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.

C. 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

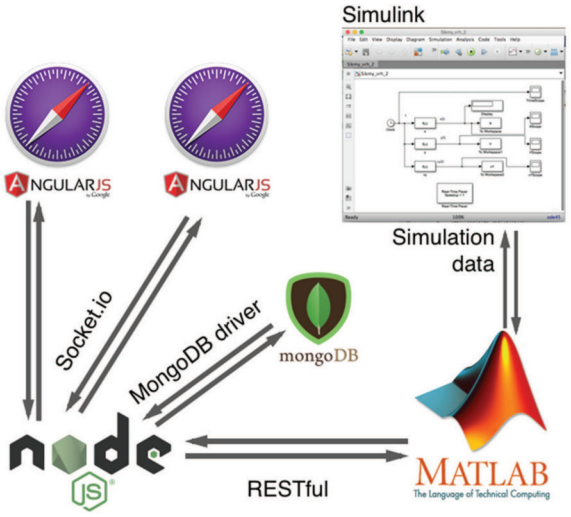

Fig. 2. Design of communication between components

- 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 [6]

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 bear in mind as properly solve the scalability, potential safety issues, multi-user access and other possible issues.

III. 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* [7] 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* [8]. Firstly, the simulation must be run through the web browser, after that data will be transferred over socket.io 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).

A. 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.

B. 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 realtime 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 *JSONlab* v1.2 [9].

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' xfine))) 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.

```
app.post('/matlab/run', function (req, res) {
    var cmd = '\/Applications\/MATLAB_R2015b.app\/bin\/matlab -nosplash -nodesktop -noFigureWindows ' +
        '-r \"cd \/Users\/Erich\/Desktop\/DP\/Matlab\/diploma-matlab\/;Sikmy_vrh_par('
        + req.body.v0 + ',' + req.body.alfa_deg + ',\'' + req.session.user + '\');projectile_sim;exit;\"';

    shell.exec(cmd, function (code, stdout, stderr) {
    });

    res.redirect('/dashboard');
});
```

Fig. 3. Start Matlab in command line using shell.js library

C. 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.

D. 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 [10] 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.

IV. Remote Control of Experiment

A. Web client created with Angular framework

Client application was created with the JavaScript framework Angular [11] (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 shows login page for web client application. It is authenticated against LDAP server of Slovak University of Technology.

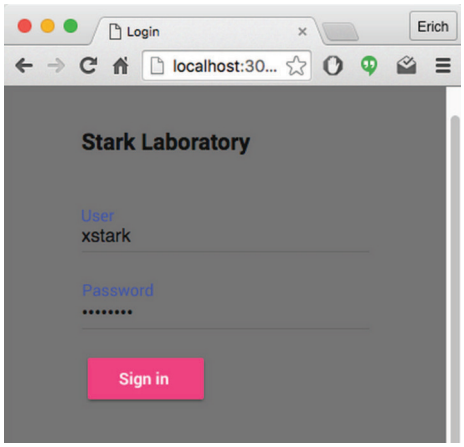

Fig. 4. Login to web application

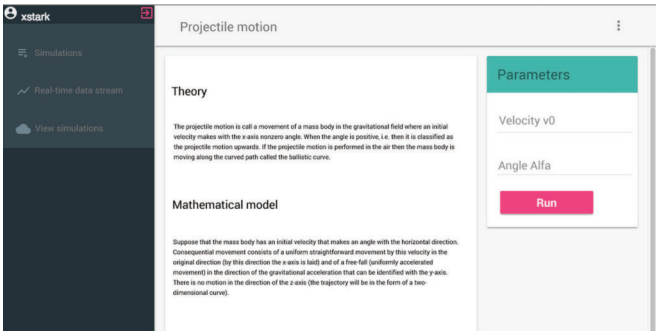

Fig. 5. Parameters of simulation / initial velocity and angle in degrees

The details of the login process via LDAP is not interesting for this part of the paper. After successful login, the dedicated page for the tested experiment is showed. Our experiment was projectile motion. It takes two parameters to run a simulation. On Fig. 5 it can be seen the form that takes two parameters to run a simulation. The page is redirected to *http://localhost/matlab* route, where the user is waiting to see the data from Node.js REST API.

It redirects to the dashboard page, and the user has to wait until the start of Matlab simulation. When it starts, the user will see new data coming to graph, animation, and table in his web browser. This part could be accelerated by a powerful server running with Matlab.

Visualization of the received data is done by Chart.js library on Fig. 7. Our implementation of chart was created using Angular directive with name *<ui-graph></ui-graph>*. Because of this approach (the usage of Angular components), it can be used multiple times with the same codebase.

In the beginning, it is necessary to get an element from DOM (Document Object Model) tree. Next step 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.

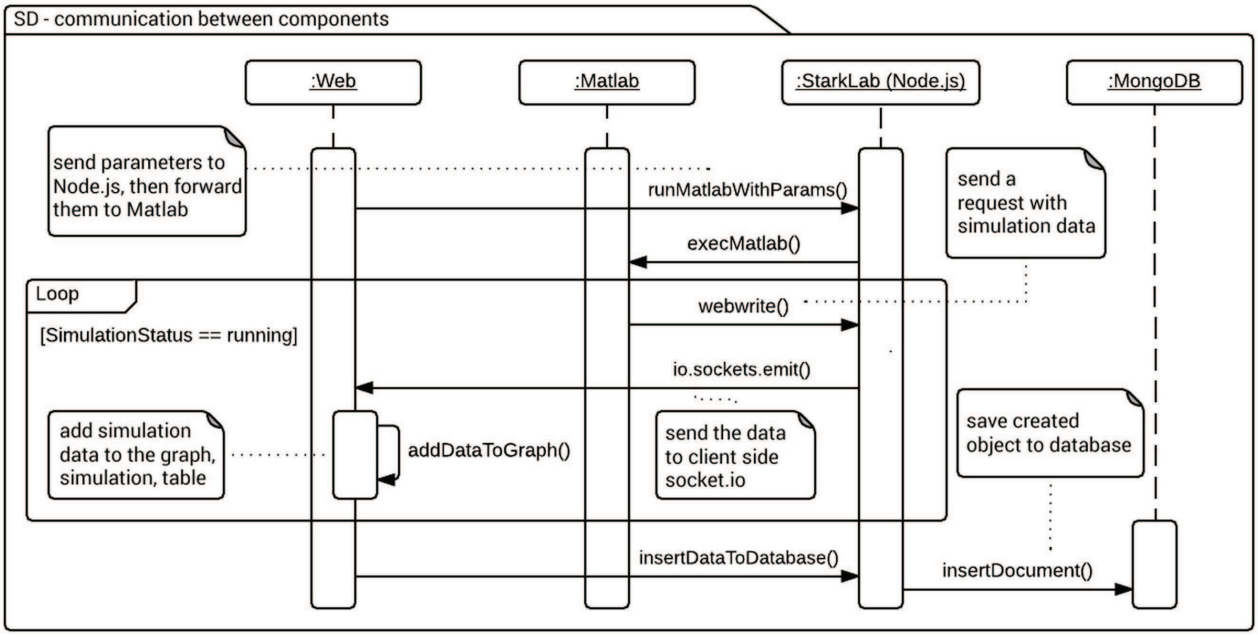

Fig. 6. Communication between components

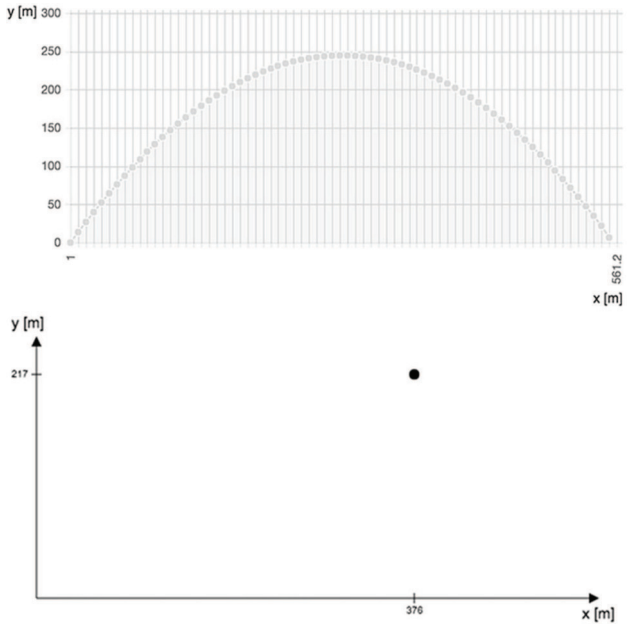

Fig. 7. Graph and Animation of projectile motion in [x, y] position

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 [11] 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

Simulation data

Time [s]	Axis x [m]	Axis y [m]	Velocity [m/s]
0	0	0	38.57
0.01	0.46	0.39	38.47
0.02	0.92	0.77	38.37
0.03	1.38	1.15	38.27
0.04	1.84	1.53	38.17
0.05	2.3	1.92	38.08
0.06	2.76	2.3	37.98
0.07	3.22	2.68	37.88
0.08	3.68	3.05	37.78
0.09	4.14	3.43	37.68
0.1	4.6	3.81	37.59
0.11	5.06	4.18	37.49
0.12	5.52	4.56	37.39
0.13	5.98	4.93	37.29
0.14	6.43	5.3	37.19
0.15	6.89	5.67	37.1
0.16	7.35	6.05	37

Fig. 8. Table data / time, x, y, vy values of projectile motion experiment

List of simulations

User	Date	Model	View	Delete
xstark	09.04.2016 11:39	projectile		
xstark	09.04.2016 11:42	projectile		
xstark	13.04.2016 09:22	projectile		
xstark	13.04.2016 09:23	projectile		
xstark	13.04.2016 09:23	projectile		
xstark	14.04.2016 02:00	projectile		
xstark	14.04.2016 02:00	projectile		
xstark	14.04.2016 02:00	projectile		
xstark	14.04.2016 11:41	projectile		
xstark	14.04.2016 11:46	projectile		
xstark	14.04.2016 11:52	projectile		
xstark	14.04.2016 11:53	projectile		
xstark	14.04.2016 11:54	projectile		

Fig. 9. Table of saved simulation for currently logged user

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. 10. 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.

V. Summary of experience with the creation of Matlab-Node.js virtual laboratory

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 the unified protocol for data interchange. Suitable would also be 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 current solution is not possible to deploy into production without certain modifications and integrations, but it might serve as a solid basis for adding new features. There are many ways to improve this solution.

All code is open source and available at https://github.com/erichstark/.

VI. 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 [12], 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.

A. Introduction to mixed reality

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

Very interesting project is a virtual clinic [13]. 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 Animech Technologies. This company offers many education modules like Virtual Car, Virtual Truck or Virtual Gearbox [14]. 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 [15], 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 [17] 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 [18] 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

Fig. 10. Microsoft HoloLens - mixed reality application (Volvo) [16]

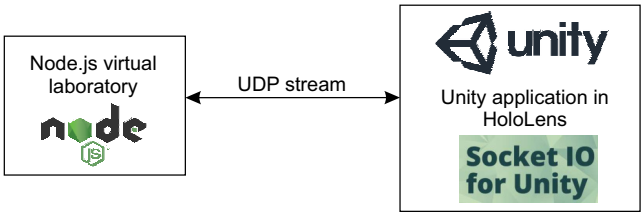

Fig. 11. Scheme - visualisation of virtual laboratory in mixed reality developed in Unity

an interaction between students that results in a significant multiplier education effect.

From technical fields there is an application called Holo-Engine [19]. 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.

B. 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 [20] which was used in the proposed application.

In Fig. 12, 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. 13.

VII. 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

Fig. 12. Results from virtual laboratory in Unity engine

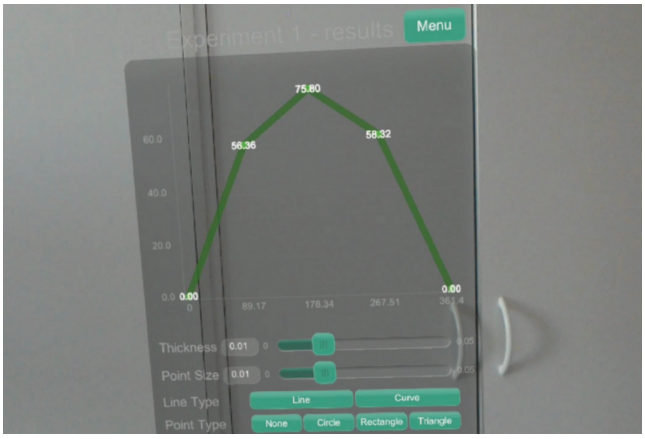

Fig. 13. Results from virtual laboratory in mixed reality (Microsoft HoloLens)

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/.

ACKNOWLEDGMENT

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-4/2015 and 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 1/0937/14 and VEGA 1/0819/17 and by the Tatra banka Foundation within the grant programme Quality of Education, project No. 2016vs046 (Support of education in mechatronics through virtual reality).

REFERENCES

[1] R. R. Wright, "Using 3 dimensional simulation in nursing education," in *43rd Biennial Convention (07 November-11 November 2015)*. STTI, 2015.

[2] V. team. (2016) The philosophy of virtual laboratories. [Online]. Available: http://vlab.co.in

[3] Z. Nedic, J. Machotka, and A. Nafalski, *Remote laboratories versus virtual and real laboratories*. IEEE, 2003, vol. 1.

[4] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez, "Remote laboratories for education and research purposes in automatic control systems," *IEEE transactions on industrial informatics*, vol. 9, no. 1, pp. 547–556, 2013.

[5] E. Stark, "Virtual laboratory using javascript on the server side (in slovak)," Master's thesis, Slovak University of Technology in Bratislava, 2016.

[6] L. Gomes and S. Bogosyan, "Current trends in remote laboratories," *IEEE Transactions on industrial electronics*, vol. 56, no. 12, pp. 4744–4756, 2009.

[7] G. Vallabha, "Real-time pacer for simulink," *The MathWorks, Inc*, vol. 21, 2010.

[8] M. team. (2017) Web access. [Online]. Available: https://www.mathworks.com/help/matlab/internet-file-access.html

[9] Q. Fang. (2016) Jsonlab: a toolbox to encode/decode json files. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files

[10] Contributors. (2017) shell.js. [Online]. Available: https://github.com/shelljs/shelljs

[11] M. Hevery and team. (2016) Angular framework. [Online]. Available: http://angularjs.org

[12] Triseum. (2017) Variant: Limits. [Online]. Available: https://triseum.com/calculus/variant/

[13] D. Parvati, W. L. Heinrichs, and Y. Patricia, "Clinispace: a multiperson 3d online immersive training environment accessible through a browser," *Medicine Meets Virtual Reality 18: NextMed*, vol. 163, p. 173, 2011.

[14] A. Technologies. (2014) Virtual gearbox. [Online]. Available: http://www.animechtechnologies.com/showcase/virtual-gearbox/

[15] P. A. Rauschnabel, A. Brem, and Y. Ro, "Augmented reality smart glasses: definition, conceptual insights, and managerial importance," *Working paper, The University of Michigan-Dearborn, Tech. Rep.*, 2015.

[16] E. Uhlemann, "Connected-vehicles applications are emerging [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 25–96, 2016.

[17] M. Corporation. (2017) Holotour. [Online]. Available: https://www.microsoft.com/en-us/hololens/apps/holotour

[18] S. Prajapati, E. Madrigal, and M. T. Friedman, "Acquisition, visualization and potential applications of 3d data in anatomic pathology," 2016. doi: 10.15190/d.2016.15

[19] 360world Europe Kft. (2016) Holoengine. [Online]. Available: https://www.microsoft.com/en-us/store/p/holoengine/9nblggh4wkh9

[20] F. Panettieri. (2014) Socket.io for unity. [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/21721