Co-funded by the European Union

SREE Scope & Features

Deliverable D3.1

Volodymyr Kazymyr
CPNU

Version 1.1
27.02.2024

Co-funded by the European Union

Record of changes

Version #	Status of document	Date	Authors	Comments
0.1	Draft version	01.02.2024	V. Kazymyr	Draft version
1.0	Full version	25.02.2024	V. Kazymyr	Consolidated version
1.1	Edited version	27.02.2024	A. Zabašta	Edited version

Quality assurance

No.	Date	Version	Approved by
1	27.02.2024	1.1	N. Kuņicina

Co-funded by the European Union

CONTENTS

1. Summary of the Report

In our time, when the countries are under the epidemic and wars, it is very important for educational institutions to provide their students with the opportunity to fully continue their work remotely. All specialties, the study of which requires the presence of certain physical equipment, encountered problems, as teachers need to find ways to provide students with all the necessary equipment for study. Those specialties that study hardware development, in particular digital circuitry using programmable logic integrated circuits and microprocessors, encountered such problem. Therefore, the question of creating such an application that will allow students to perform laboratory work online, for which they only need access to the Internet, becomes crucial.
The main objective of the project is to address the regional and specific priorities and neds of Moldavian and Ukrainian partners and to support the further reformation of Ukrainian and Moldavian HEIs according to the CBHE call priorities and ET2020 strategy of European Union.

Development of SREE within the DIGITRANS project is one of the project priorities, which is related to the tasks of WP3. ***The aim of this Report is to develop a concept of the Sharing Remote Experiment Environment (SREE) platform and to describe the scope and features of the platform.***

Objectives of SREE:

- Developing the Sharing Remote Experiment Environment (SREE) platform for on-line laboratory works with physical equipment of remote laboratories for learning and teaching practical topics in computer and electronic engineering.
- Integrating SREE with Sharing Modelling and Simulation Environment (SMSE) that affords virtual laboratories based on open software kernels using Jupiter Notebooks, for resulting acquisition and piloting of Digital Learning Ecosystem (DLE).
- Creating methodology of implementing and sharing remote applications of the HEIs laboratories' equipment and software tools for distance usage in framework of DLE based on application of ICT tools.

Component Limitation: purchased hardware for two laboratories, open-source software.

2. Project Overview

2.1. Preliminary Architecture

The general idea is to provide laboratory work using FPGA and MCU boards remotely, namely to test the operation of digital circuits implemented on FPGA or MCU, but the user only needs a PC with Internet access for this. The SREE architecture designed for this, is shown in Figure 1.

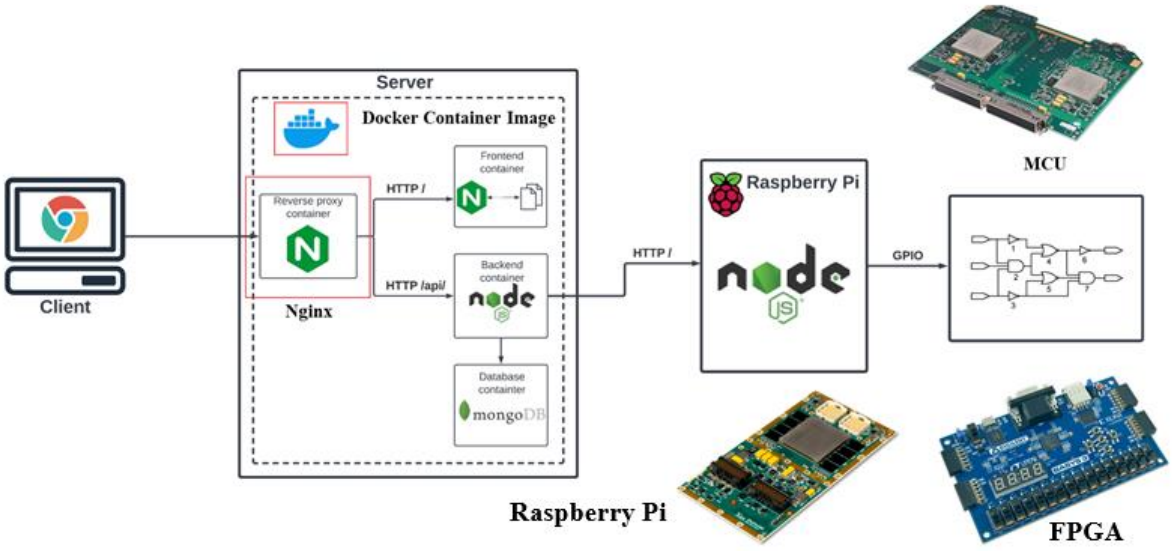

Figure 1 Suggested architecture of SREE

The main components of SREE include:
- Client PC with web interface.
- Server to connect with hardware equipment.
- Base platform to communicate with hardware boards.
- FPGA and MCU demo boards.

2.2. Component overview

2.2.1. Base platform

The basic component of SREE architecture is a hardware module capable of communicating with the FPGA board and MCU. The speed of operation, simplicity and speed of development, the cost of the device, the possibility of scaling and the cost of developing the project in the future depend on the correct choice of the appropriate platform. In order to make a choice that will the students to balance the above characteristics, they should first analyse the requirements for the hardware platform in details.
First, since the main method of signal transmission in these boards is input-output ports, the hardware module responsible for interacting with the boards must have such ports. A microcontroller, another FPGA, or a single-board computer are suitable for this.

Secondly, the platform should be able to connect with other components of the system using network interfaces. This is difficult to do on FPGAs and microcontrollers using libraries for network protocols such as LwIP, it is much easier and faster on single-board computers that can run programs written in higher-level languages than C.
Third, to scale and develop the project in the future, the platform should provide the ability to update the program code or firmware. This is quite convenient and easy to do on single-board computers running Linux operating system, which allows remote system management, such as SSH protocol, and data transfer, such as FTP protocol.
Thus, the use of a Raspberry Pi single-board computer as a hardware platform is suitable for this system, since it:

- allows you to write an algorithm in all popular programming languages, which, firstly, increases the speed of development, allows to use ready-made libraries and thus reduces the cost of development.
- works under the Linux operating system, which allows to easily update the code or transfer files (for example, log files) using special remote-control protocols.
- has more computing power than microcontrollers.

2.2.2. Programming of base platform

2.2.2.1. Working with the file system

The main tasks of the Raspberry Pi in this system are receiving and transmitting messages over the network and writing and reading signals from the demo boards using input-output ports (GPIO).
Since Rapsberry Pi works under the operating system with the Linux kernel with GPIO and it looks the same as with all other external devices, namely working with the file system.

The basic steps for working with GPIO include:

- Export pin.
- Set the direction of the foam (input or output).
- If it is an output pin, then set the appropriate levels.
- If it is an input pin, read its level.
- When the work is completed, cancel the export of the foam.

The example of Bash commands to export of pin:

```
$ echo >/sys/class/gpio/export
Setting pin #24 to output mode:
$ echo out >/sys/class/gpio/gpio24/direction
Record "1" in pin #24:
$ echo 1 >/sys/class/gpio/gpio24/value
Write "0" in pin #24:
$ echo 0 >/sys/class/gpio/gpio24/value
Canceling the export of foam #24:
$ echo 24 >/sys/class/gpio/unexport
```

2.2.2.2. Working with network messages

To create a more flexible system with developer-friendly interfaces, it is most appropriate to use the HTTP protocol, which allows to make the interface in the form of endpoints, which is a commonly used approach in modern development. This means that the chosen technology should provide convenient work with the HTTP protocol.
The criteria for choosing technology are: speed, speed and ease of development, ease of integration with other system components.
The speed of C, Python and JavaScript is more than enough for this system, the task of which is to analyse the operation of the circuit by applying signals to it. Based on the fact that the number of pins that will be used, as well as the number of clock pulses of the signals that will be applied to the input of the demo boards, is not high, so the difference in speed will not be noticeable to the user.
Therefore, considering the above and the fact that one of the criteria for choosing a language and technology is the ease of integration with other components of the system, the TypeScript language, which is compiled into JavaScript and the Node.js environment for its execution, was chosen. Using TypeScript and and other system components will allow to build a project using common types and interfaces.

2.2.3. Server component

2.2.3.1. Choosing of programming language

Since the main task of the backend is to transfer signals from the client to the Raspberry Pi, only converting one signal format to another, a technology that can work with non-blocking I/O operations is best suited for this task.
Another important criterion for choosing a technology is its popularity, as it is identical to the quality of technology support, the number of libraries and frameworks, documentation and ready-made solutions. An additional advantage of popular technologies is a large user base, that is a community, which allows to reduce the number of errors, and therefore to increase the speed of development, thanks to the use of time-tested solutions.
Important issue is the "cost" of the chosen technology. So, for example, some programming languages, such as Java or Python, require more resources from the server on which the program is executed. The reason for this is the use of separate threads to work with users.
The safety of the technology is also important. For example, the US National Security Agency stated in 2022 that the use of languages such as C and C++ is not desirable from the point of view of source software security.
In addition, considering the fact that it was decided to use the TypeScript language for the hardware part and to conduct the project in the mono repository format, the use of the same language for writing server software will allow the use of interfaces and types described in the code base of the hardware part.
Likewise, the JavaScript code execution environment is Node.js, which fully satisfies the above criteria:

- Node.js is an open source cross-platform JavaScript runtime focused on backend development.
- Node.js implements an event-driven architecture, thanks to which the programmer operates with the concepts of "event" and "event handler", which eliminates the need to manage program flows, as in certain other languages.
- Node.js is built on the V8 JavaScript Engine using bindings that link JavaScript code with the libuv library written in the C language, which implements Node.js-specific

phenomena, such as the event queue, event loop , creates "worker threads" that perform synchronous operations in parallel with the execution of JavaScript code, which provides the opportunity to use non-blocking I/O operations in the heap.

Diagram of Node.js operation is shown in Figure 2.

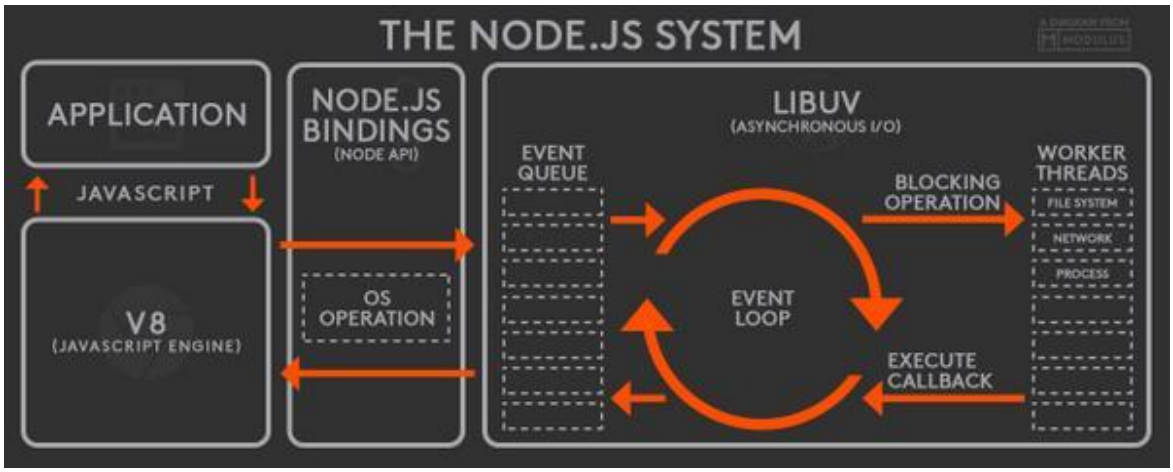

Figure 2. Diagram of Node.js operation

2.2.3.2. Choosing a framework for programming

In order to simplify the process of writing code and improve its structuring, it is advisable to use one of the popular frameworks.

One of the options is the popular framework with open source code *Express*, which is based on the use of middleware, which in practice allows to extract certain logic (for example, authorization and authentication of users, validation and transformation of data, etc.) from controllers and services into intermediate functions that will be executed before passing the execution context to the controller, thus providing an opportunity to separate the business logic from the infrastructure layer. The advantages of this framework include simplicity, speed of learning and speed of writing code. But considering the fact that the framework is not strict, the use of middlewares can significantly make the code more difficult to understand and add unnecessary side effects to the software architecture. Also, the framework does not support TypeScript by default.

The other option is the *Fastify* framework, which is faster than express, as evidenced by tests. It is also easy and fast to develop, but it has problems related to the availability of ready-made libraries.

As the best option, we suggest *Nest.js*, which is built on top of the above-mentioned frameworks (that is, during development, one can choose which of them will be used for processing requests). Its advantages are:

- It is built entirely in TypeScript language, which allows to use it much more efficiently.
- The presence of Inversion of Control (IoC), which is an important principle of object-oriented programming that allows inverting the direction of control in a program, thus increasing modularity and scalability.
- The presence of the Dependency Injection (DI) technique, which is one of the manifestations of IoC, which also increases modularity and scalability.
- It has a fairly strict style of writing code, which allows all other programmers who are familiar with this technology to understand the written code more easily.

- Support for decorators in the TypeScript language, which allows you to make the business logic code more declarative.

2.2.4. Client software

An important component of the system is the client software, which allows the user to enter signals and see the result in a convenient format. Development of a client application using a browser as an application execution environment allows to automatically make the application available from any device that supports the operation of the browser (PC, tablet, mobile phone, etc.) and saves both the developer and the user from updating problems software, because to update such application, it is enough only to place the new code on the server. Also, the language used by the browser to create dynamic pages is JavaScript, which allows to use TypeScript in all components of the project to use common types and interfaces.

2.2.4.1. Choosing a framework for programming

The three most popular frameworks and libraries nowadays are: *React, Angular and Vue*.
The peculiarity of React is that all components are created in the form of JavaScript functions or classes (the library supports both styles of writing code). However, both the code responsible for the hypertext markup and the code responsible for the logic of the application are actually in the same function (or class). This approach is called JSX. However, this library makes it quite quick and easy to create web pages at the beginning, but as the complexity of the project grows, the complexity of writing new code grows quite rapidly, because the library does not provide a strict style, and the code base later becomes difficult to understand and modify.
The Angular framework differs in that it works with the TypeScript language by default. Its feature is its rigor and approach to code distribution. So, if in React, the programmer writes logic and hypertext markup in one place, then in Angular, two separate files are used for this, one with the extension .html for writing markup, and the other - .ts for logic. For communication between components, Angular has a RxJs library that implements a development pattern called Observable.
Angular also implements the MVC model. The above-mentioned rigor is both a disadvantage and an advantage of this framework. Due to its presence, it is very popular for the development of enterprise-level solutions, since the complexity of adding new functionality when increasing the code base does not grow as fast as in React. But, at the same time, based on my own experience of developing an interface on Angular, I know that this framework requires too much effort to create applications with a complexity below the enterprise level, since the time to master this technology is high compared to analogues.
The Vue framework, like Angular, implements the MVC model. The code structure is a mix of React and Angular, as the code for the hypertext markup and the code for the component logic are in the same file, but logically separated. Thus, each component consists of two components — script (logic) and template (markup). The component files of this framework have the extension “vue”. A significant advantage of this framework is the presence of two-way bindings, which in practice means the possibility of two-way binding of elements on the web page (view) with variables in the code (model). For example, this code will make it so that when the value of the text variable is changed, the content of the input component will also change, and when the text entered manually by the user is changed, the value of the text variable will change accordingly: <input v-model="text">. Another feature of two- way

binging is support for binding also with values stored in the store, which makes the framework very convenient for writing code.
Thus, Vue is most suitable for development and will allow you to easily scale the project if necessary.

2.2.4.2. Store technology

In order to be able to exchange data between components more conveniently, third-party libraries with store technology are used, the principle of which is that data is stored separately from components, and these components, in turn, are able to read and write this data.
Two popular vue framework solutions are Vuex and Pinia.
Pinia has significant advantages over Vuex, namely:

- Simplicity.
- Modularity.
- TypeScript support.
- Availability of convenient tools, such as web browser extensions for development.

Therefore, it is proposed to use Pinia as a store for the user part.

2.2.4.3. Web-server technology

Considering that from the point of view of the browser used by the client, the code of the client application is only static files with the extension .css, .html, .js, it is necessary to choose the way in which these static files will be given from the server to the client.
The most popular solutions to this issue are Apache and Nginx.
Apache features that it is an open-source HTTP server developed by users, it uses multi-threading modules to handle user requests, and it has a multi-threaded processing model (meaning that each request is processed in a separate thread).
Unlike Apache, Nginx is an asynchronous web server and reverse-proxy, that is, requests from more than one user are processed in one thread, which allows to increase performance even under high loads. The project is also open source, but development is supported by a company with a similar name.
Because Nginx is a faster web server technology and better able to support a large number of users [C10K problem], it is recommended as a web server for serving static files to the client.

2.2.4.4. Reverse proxy

A reverse proxy is a technology that is a server that acts as an intermediary between the entire global network and the server or servers on which the web servers are running.
That is, the reverse proxy server works on the principle that it intercepts network packets coming from the Internet and, in accordance with its configuration, delivers them to the corresponding server programs.
Possible applications are:

- Load balancing - in the case of horizontal scaling of the application (when more than one application with web server functionality is running at the same time), the reverse proxy can forward requests received from users to all web servers in such a way that the load on them is approximately the same.

- Imposing restrictions on requests — a reverse proxy can, for example, reject requests that are larger than the specified size without forwarding them to the server, thus reducing the unnecessary load and relieving the programmer of the responsibility for writing similar logic in the code of the server directly.
- Acceleration of work - the reverse proxy has the ability, with the appropriate configuration, to cache responses from the server and give them to subsequent user requests, without requiring the server to process them again.
- Security - because it receives requests and replies directly to the reverse proxy, rather than the server or servers responsible for processing the request, the system is thus more private and unknown to outsiders, providing greater privacy and security.
- Solving the CORS problem — since the browser blocks cross-origin requests by default, it is necessary to make the appropriate configuration in the server code to allow the client software to send requests to the server. But, thanks to the use of a reverse proxy, which makes requests to the back-end server and the front-end server go to the same IP address and the same port, no CORS settings are necessary in such a system.

As a reverse proxy, as well as a web server for a user application, Nginx is offered as the most popular solution and, moreover, this technology is already used in the project, which makes it easier to work on the system in general.

2.2.4.5. Database

Given that the data that the user application will send to the server will come in the form of JSON, the most convenient technologies for this are document databases such as DynamoDB and MongoDB.

2.2.4.6. System deployment

To solve the issue of software deployment, containerization technology is suitable, which helps to run programs in isolated user spaces, which at the same time use the common core of the operating system

Docker is currently the most popular containerization technology which provides:

- Each component of the system will run in an isolated environment, that is, software versions will not overlap at all.
- Again, isolating each application isolates them from the software installed on the physical server.
- Considering that in this system the user needs only one port (the port that uses reverse proxy), then all other ports (database, server, web server) can be hidden from the outside environment. Containerization technologies allow you to create a network of containers that can use each other's hidden ports without exporting these ports to the outside.

In addition, with the help of containerization technologies, it is possible to solve the problem of addressing different components of each other, because within one common network for many containers, there is a DNS that allows you to use container names instead of IP addresses.

2.3. Hardware catalogue:

- Server, routing equipment – 1 set.
- Network equipment - 2 set.
- Single-board computers (Raspberry Pi 4) – 10 pcs.
- Development boards for digital design (FPGA boards) – 10 pcs;

- Development boards for microcontroller systems (MCU boards) – 10 pcs;
- Extension boards for Single-board computers – 10 pcs;
- Multi-function laboratory device (oscilloscope and ets) – 10 pcs;
- Electrical and electromechanic components – 10 sets;
- Printed circuit boards – 10 pcs;
- Webcam – 10 pcs;
- PCs – 10 set;
- Laptop – 2 pcs;
- APC Smart – 1 pcs.

2.4. Performance/Capacity

- 2 labs
- 5 simultaneously clients on every lab
- Support for at least 5 years
- 1 GB disk space for every user

2.5. Security and Safety

- Periodic database backups must be performed.
- Communications should have a secure connection. Need SSL certificates for HTTPS

3. Functional specification

3.1. SREE functions

The following features need to be released in the project based on SREE:

- Remote interaction with physical equipment via web-based interface.
- Setting up and programming demo boards for training tasks.
- Testing and evaluation board firmware.
- Visualization and live view support of remote equipment operation via webcams.

An example of one of possible hardware experiment environment is shown in Figure 3.

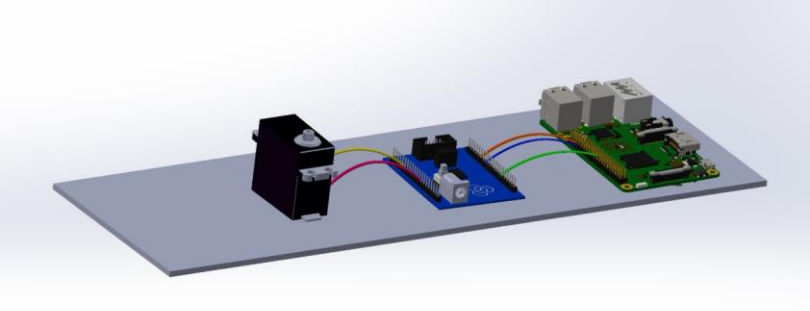

Figure 3. Right to Left: Raspberry Pi, FPGA demo board, extension boards with servo motor as a peripheral device

So, the devices are physically located next to each other on the same shelf, connected by wires. The presence of a servo motor is important, reflecting the possibility of using an FPGA or MCU to control external peripherals by feeding the appropriate signals through the Raspberry Pi. This concept allows students to combine the study of digital circuits with electromechanical devices.

3.2. Integration SREE with SMSE

This should include:

- Implementation of unified access to SREE via LMS Moodle.
- Integration SREE with SMSE to combine the theoretical and practice parts of training in updated and newly developed courses.
- Providing time slots to starting and finishing of remote experiment via Moodle.

The schema of integration SREE with SMSE is shown in Figure 4.

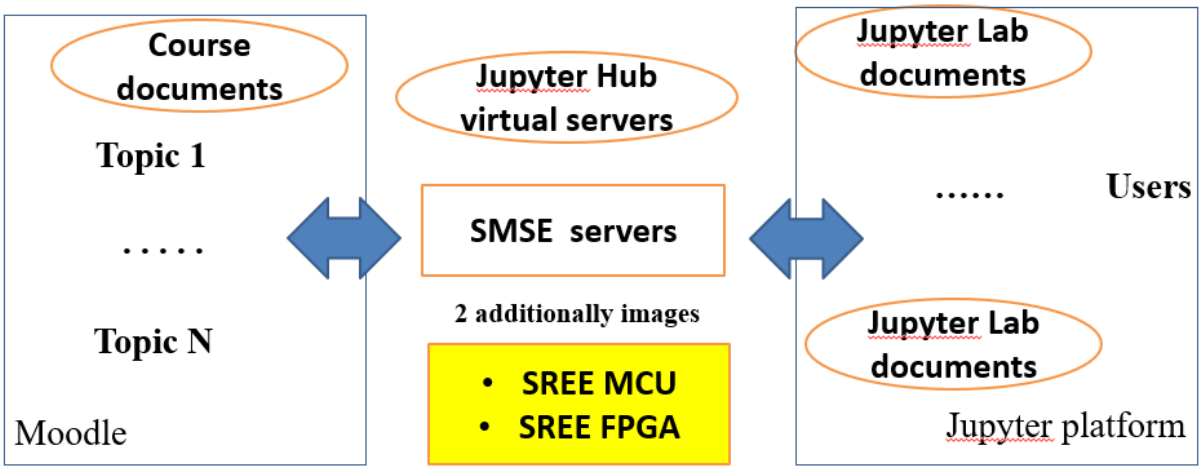

Figure 4. Schema of the integration SREE with SMSE

Thanks to integration with SMSE, the creation of remote Jupyter Labs and access to course documents in Jupyter Notebook format will be provided from Moodle. As result, a Digital Learning Ecosystem will be created (Figure 5).

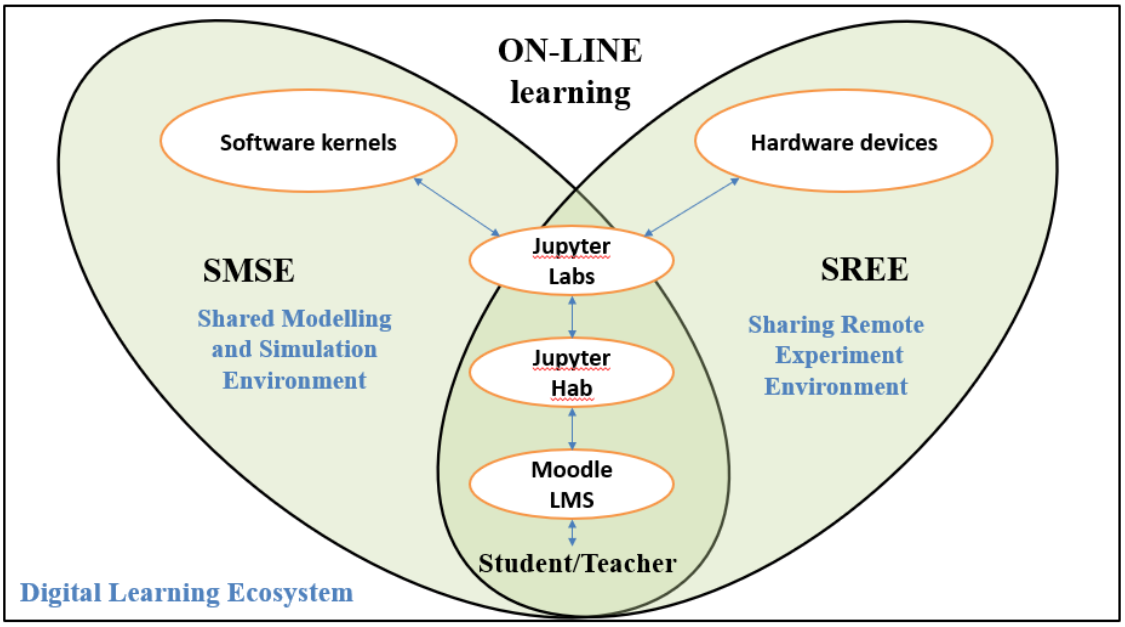

Figure 5. A concept of the Digital Learning Ecosystem

4. Interface specification

4.1. Input signals

An example of web-based interface to set up of input signals is shown in Figure 6.

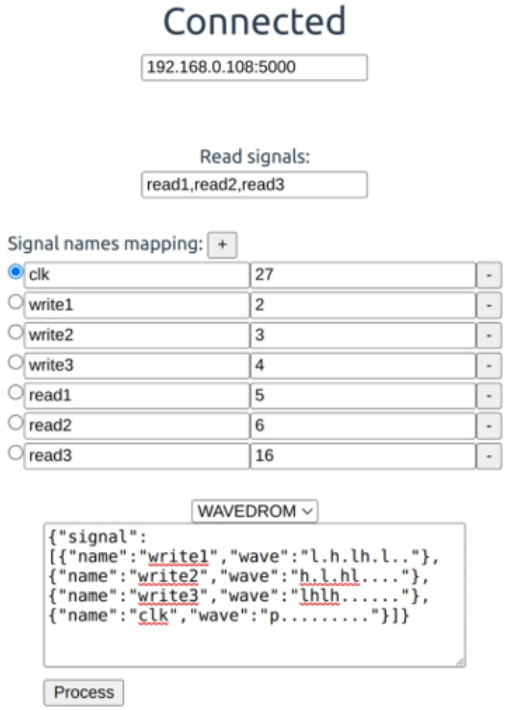

Figure 6. Web-based interface to set up of input signals

A chart will be developed to match signal names to Raspberry Pi pin numbers.

4.2. Output signals

A logic analyzer will be connected to the GPIO of the Raspberry Pi for drawing signal diagrams and their subsequent display in the web iterface as it is shown in Figure 7.

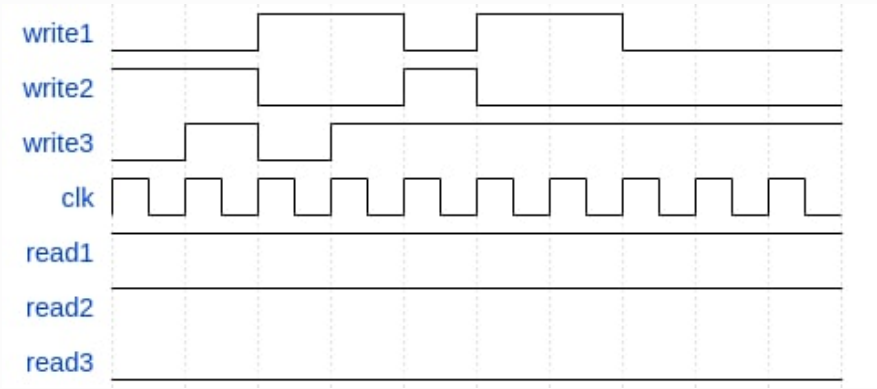

Figure 7. Signals diagram that subsequent displays in the web iterface

4.3. Main window of SREE web-interface

The main window of web-interface is shown in Figure 8.

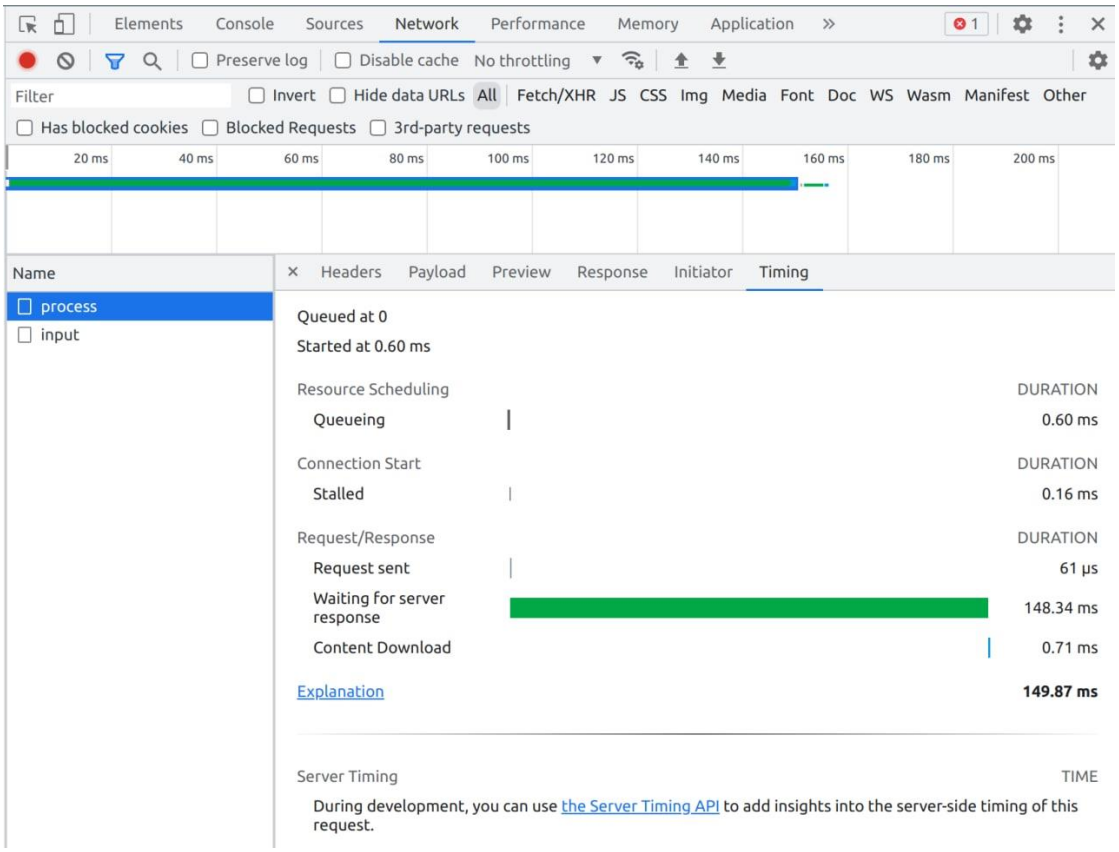

Figure 8. The main window of the web-interface

4.4. Interface for Integration with SMSE and Moodle

SREE can be launched from SMSE by use of link to SREE web-interface in Jupyter Notebooks. To launch the SREE from Moodle, the link to external resource can be used same to SMSE (see SMSE Manual from CybPhys project: https://stu.cn.ua/mizhnarodna-diyalnist/mizhnarodni-programy-ta-proekty/proyekt-cybphys/).

5. Project plan

Development of SREE includes such stages:

N	Activity	Term
1	Purchase of equipment (hardware and software)	August, 2024
2	Installation of the software on the servers of the University	November, 2024
3	Implementation of the necessary functions	February, 2025

N	Activity	Term
4	Development of SREE interface	March, 2025
5	Configuring all SREE software	June, 2025
6	Integration with SMSE including account creation	October, 2025
7	Development of methodical documents to acquisition and piloting of SREE in framework DLE	February, 2026
8	Testing of SREE	June, 2026
9	Delivery-acceptance of SREE	August, 2026

6. Project team

For the Development, testing and acquisition of SMSE we suggest the project team:

1. Project manager.
2. Programmer.
3. Engineer.
4. Tester.