Thesis Outline

Title

BUILDING A DIGITAL TWIN FROM AN INTELLIGENT ELECTRONIC DEVICE

Student

Prasath Manickam

Thesis Advisor

Prof. DI Dr. Marc Kurz

Date and Time

Hagenberg, 08.02.2021

Table of Contents

1. Overall Goal, Motivation, and Research Questions

This thesis aims to present an approach for building a digital twin solution for the increasingly complex automation problem in the Energy Industry. As we all know, industry 4.0 accelerates digitalization, smart grid, and distributed mini-grids integration enabled automation, optimization, and energy efficiency. The distributed integration and massive data led to system inconsistency, non-availability, and protection issues. Therefore, an additional infrastructure must be set up using expensive simulation devices and data for training, testing changes in protective relays, and simulations. The operators must be at the local station to check how parameters react to protection trips can be safely validated and validated with the real device and interaction. The systems must be available as a playground for the user but limited due to the machine's modeling.

The modeling of the system needs to be changed and persists with the following characteristics.

1) The operation condition of the system remains the same.

2) The system should be available as a playground remotely.

3) The core functionalities of the system are not affected, and security has adhered.

This thesis's comprehensive goal is to bring the system into browser, which replicates the capabilities of a desktop protection software SPRECON-E-P DDx6 and data is directly fed remotely. The entire system functionalities will be tested in the browser through data input, communication, testing changes, and validation.

Research Questions:

1) What is the best approach to bring Embedded Linux based system to a browser for simulation?

2) How feasible and better it works with a physical simulator (oMICRON) using Comtrade data input comparing to the local system?

2. State of the Art and Related Work

The approach to emulate Linux based system into the browser has been one of the most fascinating and challenging tasks. The present techniques QEMU and JSLinux using asm.js or WebAssembly with emscripten for getting the Embedded Linux system to the browser look encouraging, open-source, and designed for various purposes. However, a similar methodology would help us bring our system to the browser. And we will discuss related efforts and explain we focus on the method.

2.1 QEMU

QEMU is a generic and open-source system emulator and virtualizer. QEMU emulator also helps running any OS on any other OS. The most modern browsers like Chrome and Firefox can run this javascript-based emulators and makes it possible to run Linux in the browser. The javascript code is mostly slower than machine code, so boot time for the linux system would be slower. Once it boots and provides a command line interface to execute the shell commands. The emulator is written in Javascript and contains below hardware infrastructure.

a) A 32-bit x86 compatible CPU.
b) A 8254 Programmable Interrupt Timer.
c) A 8259 Programmable Interrupt Controller.
d) A 16450 UART.

It has a predefined kernel configuration and uses Buildroot and Busybox as the filesystem.

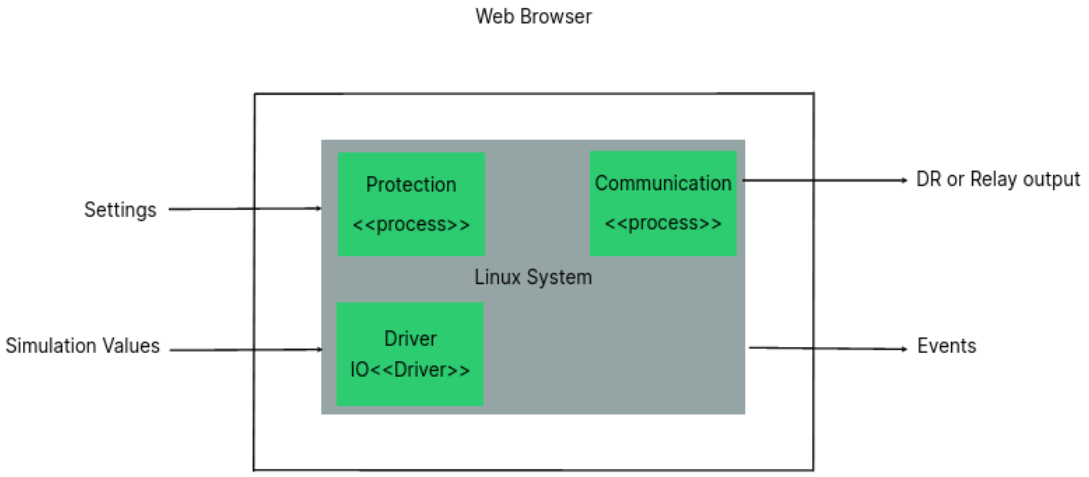
2.2 JSLinux

JSLinux is an emulator based on TinyEMU and compiled into Javascript or WebAssembly with emscripten compiler was developed in 2011. It used QEMU parts of it for x86 helpers and devices. It was first of the PC with an x86 version emulator compiled in Javascript runs Linux. Later, it was modified to use asm.js API subsets for faster browser support. It also used emscripten compiler to convert asm.js to C code and back to javascript for many functionalities addition. It added VirtIO 9P filesystem for the import or export options in the browser to add remote filesystems. The entire system was up and running in the browser except for missing few devices (keyboard, mouse, IDE disk ,and VGA). It uses Buildroot and Alpine Linux distributions. The network access is enabled inside the emulator using Websockets VPN.

Both approaches achieved to bring the Linux system into the browser with various functionalities and serves different purposes, and this work is a good start for my problem solution. We will use a similar approach to emulate a system with required processes, drivers, and I/O for our problem with an addition of wasm machine code specific to our need.

3. Approach and Methodology

The method here is that to bring Linux based system which contains application software (SPRECON-E-P DD..6 series), processes, drivers, etc. It performs monitoring and controlling the distance protection relays, combined protection, and control devices connected in series to ensure the safety of primary equipment in the medium and high voltage levels. The software takes input data from the simulation device and renders it in the UI for Web browser management.



The initial step is to find the feasibility of emulating the system into the browser through WebAssembly language using Emscripten compiler. The successful approach will be taken further to check essential communication, monitoring, and control procedures. Furthermore, the below parameters are verified to ensure the whole system.

- Events are notified and protection are ensured.
- Simulation behaviour to events in accord the desktop application works correctly.
- The accurate data inputs, outputs, communication and system performance are ensured.

4. Practical Realization

This thesis is about building a digital twin of the system(Linux) that can be used for training to test changes and simulations. This twin system will be integrated into the Web engineering replicates behavior model of the IED and executed on the web using WebAssembly. WebAssembly is a promising innovative approach that enables running machine code on the browser besides Javascript and HTML. The entire model consists of several C/C ++ programs based on RTOS and Linux.

The digital twin facilitates the system as a playground in the browser and provides an intuitive experience to testers, developers, and operators. Therefore, it helps to reduce the need for costing operators and physical availability. Also, the complex infrastructure can be simulated and tested remotely available to integrate into production devices safely.

In result, the entire system runs in the browser with limited exceptions and behaves as local desktop sytem. In addition to that, the system performance, user experience, system integration, hardware usage, internet usage, updates, security are evaluated in tabular form, and graph illustration.

5. Planned Structure of the Thesis

1. Title Page
2. Abstract
3. Table of contents
4. Introduction
5. Related Work
6. Approach and Methodology
7. Result
8. Conclusion
9. List of references

6. Planned Schedule and Deliverables

Milestone	Estimated date	Deliverable/Outputs	Standing
Project proposal	Dec 2020	Project topic acceptance	Completed
Project Specification and Thesis outline	Jan 2021	Project specification and Thesis outline	WIP
Data collection and analysis	Jan 2021	Access Data	WIP
Implementation	End of Feb 2021	Developed and Working model	Not yet started
Evaluation and research	End of Feb 2021	Research on other best approach	Not yet started
Project Thesis Writing and presentation defence	25.05.2021	Final edition of thesis written	Not yet started
	21.06.2021	Final report submission	Not yet started
	12.07.2021	Defence	Not yet started

7. Literature

[SA21] Sprecher Automation GmbH: “SPRECON Protection Device website” [Online]. Available: https://www.sprecher-automation.com/en/products/sprecon-protection-devices/ [Accessed: 02-Feb-2021]

[WA21] WebAssembly: “WebAssembly open source community website” [Online]. Available: https://webassembly.org/ [Accessed: 02-Feb-2021]

[EM21] Emscripten: “ Emscripten open source community website” [Online]. Available: https://emscripten.org/ [Accessed: 02-Feb-2021]

[QM21] QEMU: “QEMU open source community website” [Online]. Available: https://www.qemu.org/ [Accessed: 02-Feb-2021]

[JS21] JSLinux: “JSLinux open source community website” [Online]. Available: https://bellard.org/jslinux/ [Accessed: 02-Feb-2021]