Hindawi
Mobile Information Systems
Volume 2021, Article ID 6668843, 13 pages
https://doi.org/10.1155/2021/6668843

Research Article

Web-Based Human-Machine Interfaces of Industrial Controllers in Single-Page Applications

Shyr-Long Jeng,[1] **Wei-Hua Chieng**,[2] **and Yi Chen**[2]

[1]*Department of Mechanical Engineering, Lunghwa University of Science and Technology, Taoyuan 333326, Taiwan*
[2]*Department of Mechanical Engineering, National Chiao Tung University, Hsinchu 300, Taiwan*

Correspondence should be addressed to Shyr-Long Jeng; aetsl@gm.lhu.edu.tw

Received 31 October 2020; Revised 12 March 2021; Accepted 19 March 2021; Published 7 April 2021

Academic Editor: Hsu-Yang Kung

Advances in conventional industrial controllers have led to new technologies such as multilanguage use, cross-platform applications, and remote monitoring and control. However, the human-machine interfaces (HMIs) of conventional industrial controllers and mobile devices cannot directly transmit instant messages to each other. This study describes a simple method of upgrading the HMIs of conventional industrial controllers into controllers capable of Web-based remote access. The study began with the development of a model-view-controller architecture consisting of Hypertext Markup Language, Cascading Style Sheets, and JavaScript and proceeded to the implementation of a single-page application (SPA) method through AJAX and WebSocket, which communicates with the back-end Node.js server to transfer data. Future advancements will enable information to flow through cross-platform devices across various operating systems and Web browsers, allowing users to remotely monitor and control machines from mobile smart devices. We demonstrated the simplicity of the SPA method by transforming a conventional personal computer-based industrial controller, WINPC32, to an all-purpose Web-based HMI for industrial use with the graphic user interface software, GPX.

1. Introduction

With the advancement of the Internet of Things (IoT) and changes in industrial demand, the platform used for the monitoring of many industrial applications has gradually shifted to web pages and mobile devices [1, 2]. Enterprises have developed mobile apps that enable communication among their various devices while human-machine interfaces (HMIs) are important tools for industrial monitoring. The motivation behind this study was to offer a solution that allows the HMIs of traditional industrial controllers to be executed on mobile devices. The challenge was that HMIs require cross-platform functions to be executable on different operating systems (Windows, iOS, and Linux) and devices (desktops, laptops, tablets, and smartphones). One aesthetically satisfying solution is to embed the HMI of the monitoring system on a browser-based web page and to use Extensible Markup Language (XML) as the format of the system configuration file.

The objective of this study was to describe a form of Web-based remote accessibility for conventional industrial controllers that connect supervisory control and data acquisition (SCADA) control with programmable logic controllers (PLCs). Phuyal et al. [3] proposed a Web-based remote-access real-time laboratory using SCADA control. They used a PLC to control the operation of the system and installed a SCADA system to monitor and control the process. The Web interface was designed in Visual Studio with ASP.NET and allowed students to access the lab and information regarding the experiment. The remote laboratory allowed for users to control an induction motor; this was used as an example to demonstrate its effectiveness. Kondratenko et al. [4] proposed monitoring and automating control processes in specialized pyrolysis complexes (SPCs) to utilize municipal polymeric waste. They described the functional structure and the main components of their generalized SPC Web SCADA system. They also presented an example of the application they proposed for the SPC

Web SCADA system. Rather than focusing on the previously mentioned applications, we focused on industrial controllers that are top-tier SCADA and PLC devices arranged in a typical structure: an industrial controller, SCADA, and PLC connected via Modbus. Industrial controllers with an open platform communications (OPCs) server and client-based architecture use the Internet, whereas conventional controllers without OPC capabilities are standalone and detached from the Internet. Qasim et al. [5] described a model of an HMI for Android mobile devices that uses model-to-text transformation to take the domain model as an input model for generating the complete mobile Android HMI code and uses Widgets Designer (.axml) code for applications. Caiza et al. [6] proposed a Web platform for creating HMIs that uses low-cost devices to integrate shop floor information through the OPC unified architecture (UA) protocol and PLC-based automation. These works, which arose in contexts similar to that of this study, have exhibited a simple path for a conventional controller to enter Web-based HMIs. XML and JavaScript Object Notation (JSON) enable information to flow through cross-platform devices on a variety of operating systems and Web browsers.

This study used a single-page application (SPA) architecture [7] to transmit conventional industrial controller information across various cross-platform devices, operating systems, and browsers. The Asynchronous JavaScript and XML (AJAX) can also be used to implement multiple user usage scenarios. Re-encoding conventional industrial controller data into XML helps reduce file size, improves system performance, and enhances the ease of maintenance, and such information stream can be combined with browser-style Web applications to increase system development efficiency. Remote monitoring through the Internet allows users to instantly control on-site system equipment from a distance. This method can substantially reduce the cost of equipment and labor as well as the time required to convert conventional industrial controllers into Web-based controllers.

The first academic article addressing SPAs and single-page Web applications was a technical report by Mesbah and Deursen [8] in which the authors responded to a problem Web technology faced at the time: "Web applications have suffered from poor interactivity and responsiveness towards end-users." The reason for the problem was that traditional Web design relied on postbacks, which required that pages be continuously refreshed and divided websites into several different design structures. This type of design pattern is called a "multipage application." Several of the multiuser applications had obvious flaws, such as low immediacy, slowness, poor Web interaction, and poor user experience. The SPA design pattern was proposed as a solution to these problems. The shift from multipage to single-page design did not merely reduce multiple pages to a single page but provided users with an experience akin to desktop applications by moving the user interface (UI) from the server side to the client side and implementing application logic on the client side [9, 10].

The concept of SPAs arose from the popularity of AJAX technology. Technologies such as jQuery and Bootstrap and front-end frameworks such as React.js, Vue.js, and Angular.js can be used to implement single-page design. jQuery is a minimalist system primarily used to manipulate existing Document Object Model structures. IBM Dojo's package system simplifies the management of large-scale HMI development projects and incorporates high-performance implementations of common utilities into its core. Whereas Dojo uses standard Cascading Style Sheets (CSS) 3 queries, jQuery features a hybrid XML Path (XPath)-CSS query language and offers a wide range of options and operations for the results of these queries. jQuery incorporates AJAX, effects, and other utilities into its small core. Useful benchmarks for several packages were described in [11]. With respect to user experience, front-end development has seen a trend of UIs that operate similarly to desktop applications [12, 13]. The websites most frequently developed with SPA design are Web-managed web pages, in which the UI is usually a dashboard with top or side content navigation bars. Web pages, Google's Gmail, Google Drive, and Azure Portal are examples of SPAs.

In data management, graphical monitoring interfaces present information more effectively than digitally based interfaces. Chynal and Sobecki [14] analyzed users' visual focus to understand the strengths and flaws of several current web pages and suggested effective methods for developers to present data. For example, line and waveform graphs are intuitive methods for presenting changes in the voltage of a sensor, and supplementing them with pie charts or bar graphs can allow users to see statistics from a certain period of time at a glance. These graphical effects can be achieved through Web applications. These methods make quantitative data easy to interpret and more aesthetically pleasing and accurate than the old, unwieldy interfaces of traditional hardware displays [15].

The model-view-controller (MVC) software design pattern has become the mainstream Web design architecture. MVC prioritizes separating concerns and clearly distinguishing the roles of data, display, and logic, which is a particularly crucial part of HMI development. MVC can also simplify large programs into small modules or objects, making programs easily reusable, a feature which accommodates the design and development of modern SCADA-HMI systems [16]. Wang [17] analyzed two approaches to data serializing used in Web applications: XML and JSON. The results revealed that the transmission of Web application data was secure and powerful in the XML approach and fast and convenient in the JSON approach. Nurzhan et al. [18] compared two formats for data interchange currently used for industrial applications. The results revealed that JSON was faster and used fewer resources than its counterpart, XML. With the evolution of data lightweighting technology, the XML format has been replaced by the JSON format for data transmission, not only reducing the amount of data being transmitted but also increasing the transmission speed and improving the performance of the web page. In addition to MVC, a remote monitoring system enables developers to efficiently divide work, ensuring that the development of business logic and UIs remains separate.

This method was used to develop the Web application for the IoT-based toll system [19].

The component system approach subdivides an application into multiple components to construct an entire application or web page, with a component as the smallest unit. Development frameworks based on component systems have become a trend in modern Web development technology; examples are Java-based Web development frameworks such as Tapestry, Jakarta Server Faces, and Wicket [20]. Web front-end frameworks based on JavaScript include Vue.js, Angular.js, React.js, and Backbone.js. [21, 22]. The biggest advantage of component systems is that each component can be modularized and reused to achieve separation, allowing each component to perform its respective duties. Web applications built with component systems are easy to maintain, and developers can easily add new features or components to a template. Papcun et al. [23] described the evolution of HMIs and provided examples of solutions and case studies for each stage of this evolution. HMI 2.0 remote visualization and control on personal computers (PCs) can only work with networking. HMI 3.0 was applied to Internet connectivity and Web servers in companies, thereby opening the possibilities of Web and mobile applications. The integration of easy-to-access and open-source software, such as AJAX and the Node.js server, enables conventional industrial controllers to increase IoT connectivity.

After substantial development, the IoT has come to play the vital role of enforcing security and privacy policies to avoid a variety of vulnerabilities and threats to systems. IoT security requirements are classified into the following categories: authentication, access control, maintainability, resilience, data security and data sharing, security monitoring, and network security [24]. One approach to security is to use a virtual private network (VPN)—a secure connection between a device and a VPN server over the Internet. VPN private networks create secure communication channels between two points over public networks [25–27]. Because OPC UA did not use a VPN, its ability to provide security for industrial process control systems was compromised. Hackers obtained usernames and passwords and gained access to the system. OPC UA servers [28] were then outfitted with additional security features, such as server certificates for signing and encryption, authentication certificates for logging in, and a trust-reject function for controlling access (signing procedure) and identifying who can connect to the server.

In this study, we developed a general-purpose industrial controller HMI for browsers. Figure 1 displays the controller's architecture. We used an SPA design based on an MVC architecture and a component system using Vue.js to remotely monitor the industrial controllers and created a back-end server based on AJAX, WebSocket applications, and Node.js to manage data transmissions. The industrial controller developed in this study is based on the WINPC32 (Windows-Based Programmable Controller) industrial controller software, and the results of the development output by WINPC32's graphic control software, GPX, were converted into an SPA by a parser. The controller allows users to achieve cross-platform remote monitoring. By transforming a conventional PC-based industrial controller with the WINPC32 graphical user interface (GUI) software, GPX, into a general-purpose Web-based HMI for industrial use, we demonstrated the simplicity of the SPA methodology. The cyber-physical systems of Industry 4.0 [29] can be implemented to perform remote diagnoses or collect data for machine learning through the Internet.

1.1. Development of SPA Architecture

1.1.1. WINPC32 and GPX. As Figure 2 shows, WINPC32 is a programmable control and HMI platform developed by Hurco Automation Ltd. [30]. Built on this foundation is a suite of soft-programmable industrial-control modules that encompass Soft-PLC, process PID, Soft-Motion, industrial networking, and machine vision. The manner in which the remote data included in the WINPC32 system are implemented is by the SCADA engine through communication drivers or OPC UA servers via communication master cards or direct serial link through COM or Ethernet ports once the input and output (I/O) is configured. On top of those real-time soft modules is an HMI module, which provides advanced HMI functions and network connectivity to Web, LAN, the major field bus, and PLC controls. The Soft-PLC complies with IEC61131-3 and offers the main types of PLC editing languages. The programming environment of all other modules, such as motion and HMI, is also compatible with major industry standards. The hardware I/O supported in WINPC32 includes the local ones assigned from various PC-Based Add-On cards, including a digital I/O module, analog I/O module, process proportional-integral-derivative (PID) interface module, motion interface module for motion, and data acquisition module, and remote ones supported from the various field bus slave I/O and device nodes or controls like PLC or temperature controller. Soft-PLC Editor offers three major PLC programming languages: Ladder Diagram, Instruction List, Function Block (FB), and Structured Text. In addition, WINPC32 also offers standard C language for logic programming. The C programming can be used for partial (just a customized function) or entire logic controls. PID software configures the I/O of each PID loop corresponding to I/O or other variables. In addition, PID software establishes control parameters, such as PID gains for on-off, or proprietary control loops. GPX is a visualization development tool with which to edit GUI and recipes, program flow control logic, configure alarm logs, and establish security and remote monitoring. The run-time engine includes the Global Data Exchange Server (GDS). The GDS is responsible for executing the hardware and software modules the user has selected and configured as a hybrid system at a predefined priority and timing.

The GPX Windows graphics control software is an HMI-editing software for the PC-based general-purpose industrial controller WINPC32. With its integration and application of WINPC32 and various control modules, GPX offers a wide range of applications for both traditional and high-tech industries.

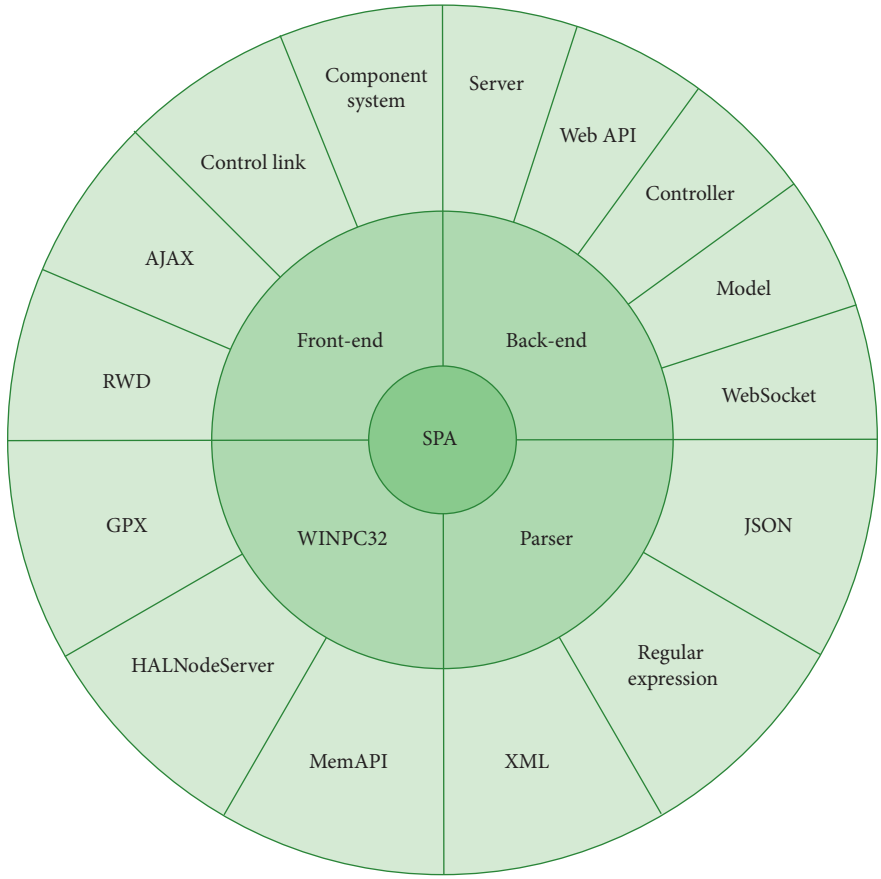


Figure 1: Architecture of the general-purpose Web-Based HMI for an industrial controller.

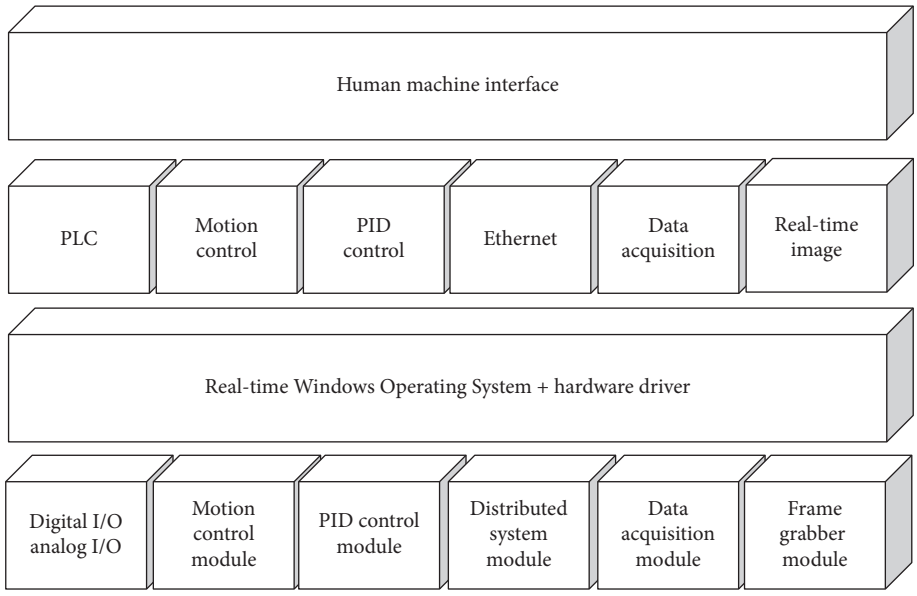


Figure 2: The system architecture of WINPC32.

GPX contains two main programs: Windows Maker for developing the controller HMI and Windows Viewer for presenting the results. Developers use GPX's graphical interface to develop and design HMIs because the program makes it easy to adjust the component control style and set the memory address mapped to the hardware. After the development phase, GPX exports the files in XML. Figure 3 shows an example of a project created with GPX. The main

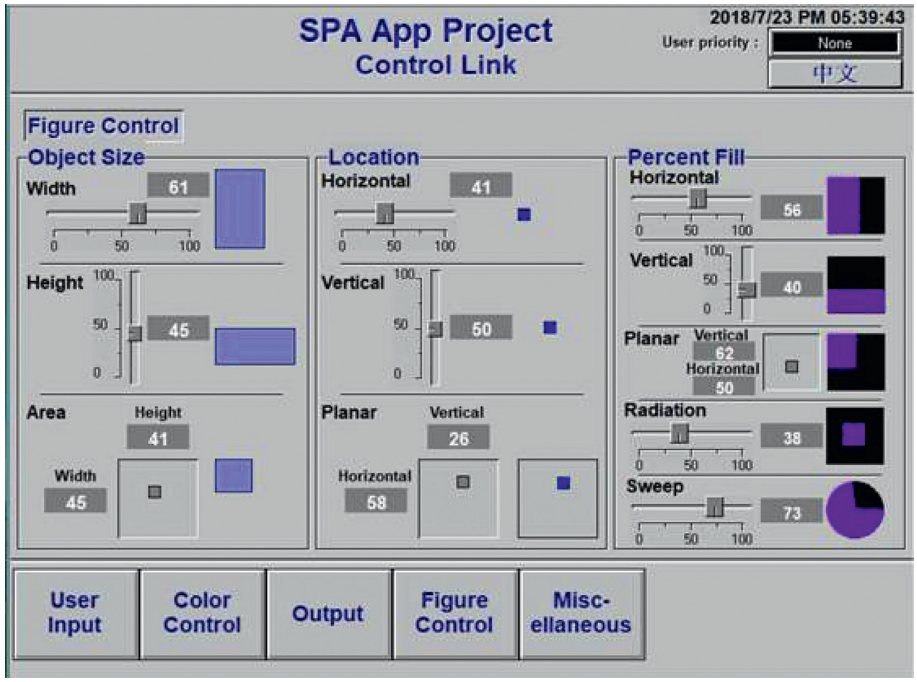


Figure 3: SPA project in GPX.

purpose of this study was to translate the data from the original desktop application to the Web application executed on the browser in an XML file.

1.2. Front-End and Back-End. As Web technology has progressed, website functions have become increasingly complex. The structure of web pages increasingly tends toward the division of responsibilities and designs that separate concerns, as in the MVC architecture. To improve user experience, several developers have developed their own Web applications for their websites. SPAs are widely used in websites that require real-time interaction with users and several modern Web technologies, such as AJAX, WebSocket, Webpack, and the Web application programming interface (API), to run efficiently. Large-scale Web projects often require a team of developers.

To equally distribute labor, the architecture of modern Web development is usually divided into two parts: the front-end and back-end. Front-end engineers are mainly responsible for the appearance of user pages in the browser, including the web page style, page layout, user experience, and application functions. Back-end engineers are mainly responsible for server-side data access, database management, API design, website analysis, and website optimization.

The main languages for front-end Web development are Hypertext Markup Language (HTML), CSS, and JavaScript. Depending on the situation, developers can use jQuery, Bootstrap, or other extension libraries and front-end Web development frameworks such as Angular.js, React.js, and Vue.js to create a website. Modular and component-based development makes programs not only easy to build but also easy to maintain. The range of options for back-end server languages is broad. Developers can use ASP.NET C#, Visual Basic, Java, PHP: Hypertext Preprocessor, Python, Ruby, and Go, among many others. If developers use Node.js as the server, they can use JavaScript for the project's architecture to develop both the front-end and the back-end.

1.3. MVC Architecture. The components of MVC are as follows: "model" refers to data related to the application's business logic and processing of the data and can directly access data in locations such as databases. "View" represents the display logic and the rendering of the page. "Controller" represents control of the flow of Web applications and response to the various events related to user behavior and changes to the model.

Figure 4 displays a diagram of the network architecture in an MVC design. Operations are executed in the following steps: (1) the client uses the browser to send a request, (2) the server accepts the request and sends the command to the corresponding controller, (3) and (4) the controller accesses the model data, (5) the controller sends the data to the view to render the display page, (6) the viewer renders the rendered page back to the controller in HTML, (7) the controller encapsulates and generates the response data to the browser, and (8) the final display page is returned to the browser by the server.

Without dividing the front-end and the back-end, the architecture of the MVC design displays page contents by using server-side rendering to dynamically update the web page through ASP.NET MVC and Ruby on Rails, meaning that a combination of data access and page demonstration is performed on the server side. Each request by the client is regarded as an action, and the method corresponding to a page is generated by the controller.

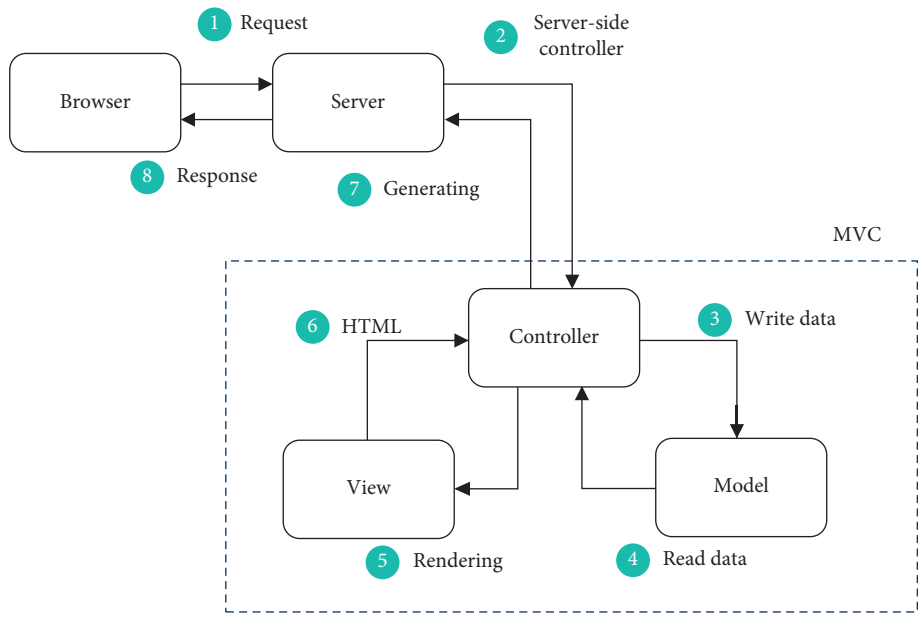


Figure 4: Web development architecture of MVC.

Although logic and data can be separated in this structure, labor responsibilities cannot be easily allocated because the engineers responsible for data access and UI design must complete the entire project to debug and develop the program—the engineer responsible for the UI must set up the database and the server. For this reason, the division of labor in this structure becomes unclear.

1.4. Client-Side Rendering. In practice, the view must be extracted from the MVC on the server side. The extracted view is processed by the client's browser: the page and the data are combined in the browser, a process known as "client-side rendering." As a result, the server focuses on returning data instead of returning the entire page's HTML file to the browser to separate the front-end and the back-end. This structure represents the design of modern web page architecture separating concerns and duties.

Making the server focus on returning data instead of returning the entire page's HTML file to the browser is common practice in modern Web design to separate duties between the front-end and the back-end. However, this approach increases the complexity of the front-end project. Figure 5 displays the web page architecture rendered by the client.

Another simple method can be implemented into the front-end as part of the "view" section of the MVC. In this method, a router processes routing problems on the front-end. The front-end route is usually distinguished from the back-end route by the "#" symbol, as in the uniform Resource Identifier, "http://localhost/#/Home/Gpx." The corresponding controller handles the flow of data, and the model requests the data from the server through the back-end API. The server returns a response containing only data and a few headers and then temporarily stores the data in the front-end, performs logical operations, and submits a display logic of the view to render the final view to the user. Although front-end engineering has become complicated, it greatly improves the separation of duties, program maintenance, and development flexibility and reduces the resource burden on the server at high traffic.

1.5. Component Systems. Component systems are an integral part of modern Web design. Component systems begin with the root component. Then, countless reusable subcomponents can be used to construct an entire web page, which can be abstracted into a component tree with trunks and branches. Figure 6 displays a tree diagram of the component system. The relationship between the components can be easily identified as parent-child or sibling-sibling. Component systems can be implemented without special development tools. Component systems are commonly implemented by using a front-end framework, such as Angular.js, React.js, or Vue.js. The development of front-end projects is modular and supports front-end routing and state management functions that make the development of SPAs smoother. In this study, we used Vue.js to develop the component system.

When using Vue.js, the data are transferred between the parent and child components through event emitting and props passing. Vuex can use a public interface to call each component in the parent layers, thereby managing the state of each component in the entire web page. Data can be transferred between sibling components through the $root.$emit () and $root.$on () functions in the Vue root component, as shown in Figure 7.

1.6. Software Development. The concept of regular expression in computer science is a key tool for writing parser programs. Regular expressions use a single string to describe a series of strings that conform to certain syntax rules. In several text editors, regular expressions are often used to

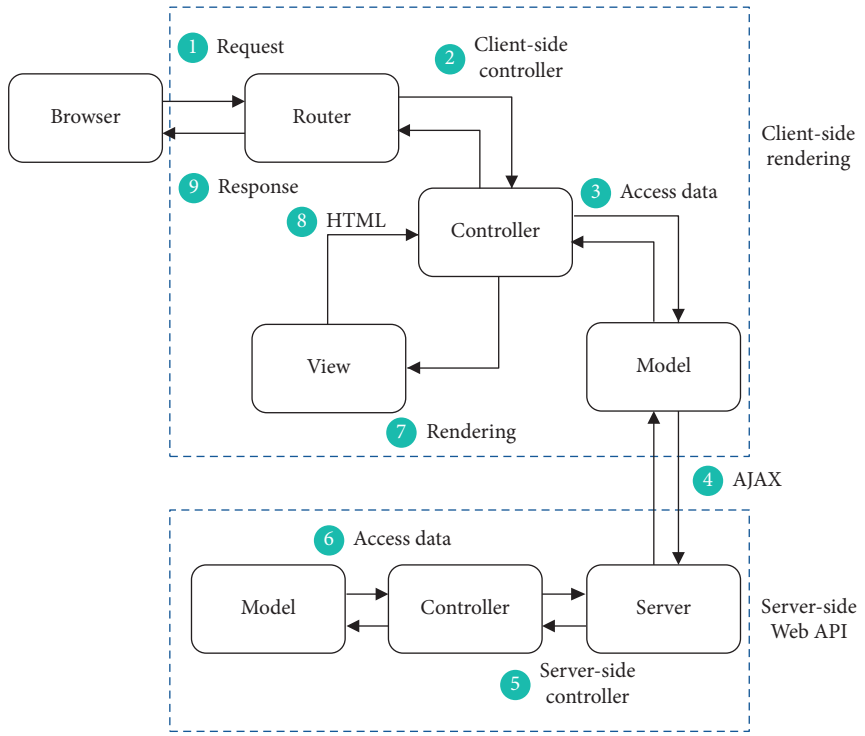


Figure 5: MVC architecture with client-side rendering.

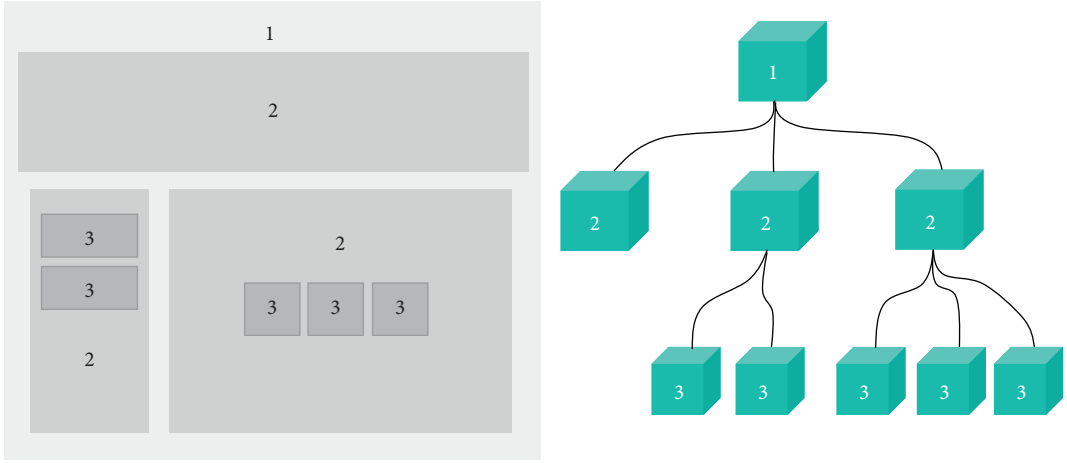


Figure 6: Tree diagram of the component system.

retrieve, filter, and replace text that matches a specific pattern. In this study, we used regular expressions to filter out the placement of individual objects, the members of each object, and the real-time data. The data could then be stored in the GPX object generated by JavaScript. When the data had been extracted, the GPX object was output as a module.exports object, allowing it to be imported into other JavaScript programs, such as server.js; by compiling GPX into JSON and defining the Web API, the front-end can access the data at any time. Figure 8 presents the main block diagram of the parser.

Two main tasks were involved in the implementation of the front-end project. The first task was to design and convert the GPX object from the back-end into various components that could interact with the Web users and completely transplant the GPX HMI to the web page. The second task was to realize the communication interface between the control elements and the AJAX Hypertext Transfer Protocol (HTTP) request. The request sent to the back-end is wrapped in the event handler of each component and is triggered by the event to execute the user's command.

The implementation of the back-end project also had two main goals. The first goal was to compile the output object of the GPX parser into JSON data and send them to each client to render the GPX style. The second goal was to design a Web API to release the data access capability of WINPC32.

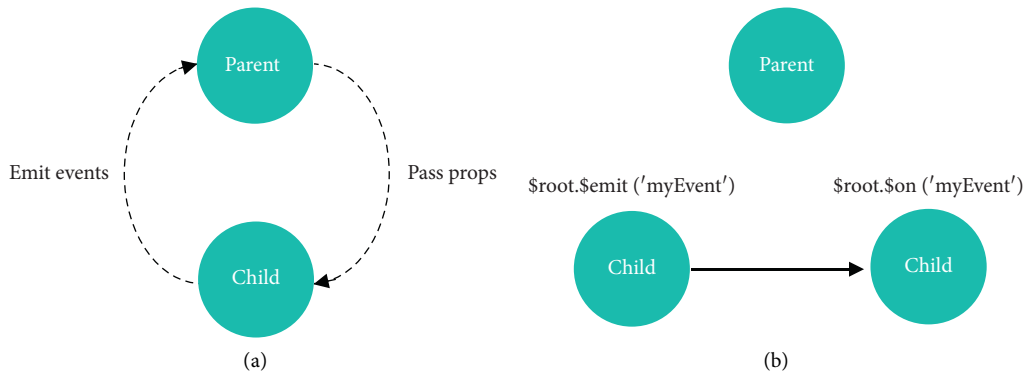


Figure 7: Communication among components. (a) Parent-child and (b) sibling-sibling.

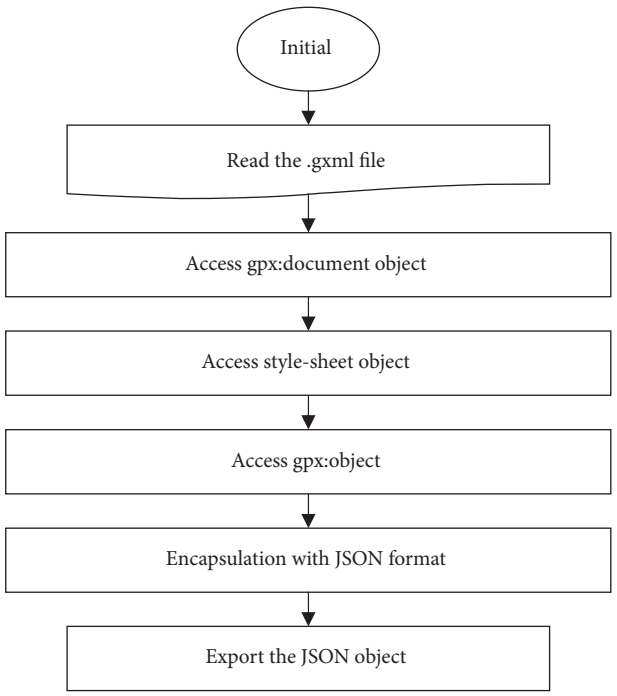


Figure 8: Main block diagram of the parser.

The API uses ClientSubscription() and ServerPublish() functions to execute industrial functions so that front-end components can call the controller to run the corresponding processing.

2. Results

Using WINPC32, we implemented the SPA of the GPX window graphics control software, converted XML files into JavaScript objects for web development, implemented object data as the HMI of the Web application, and executed different applications with the back-end WIMPC32 API. The process consisted of three parts:

(1) GPX Parser: converted XML data of GPX objects into JSON for Web development.

(2) Front-end tasks: the front-end involved two main tasks. The first task was to convert the GPX objects from the back-end into the web page. The second task was to implement the communication interface for the control components to send AJAX HTTP requests to the back-end. The communication interface was wrapped in the event handler for each component and triggered events to execute the functions of the HMI. The user input command was then sent to the back-end.

(3) Back-end tasks: the back-end project involved two main tasks. The first task was to compile the output objects of the GPX parser into JSON data and send them to each client to render the GPX style. The second was to design a Web API that integrated WINPC32. The data access function in the front-end control component called the controller for function processing in the form of API to achieve industrial control.

The GPX application project, depicted in Figure 3, was used as the input source for the implementation software shown in Figure 9(a). The reason for selecting this project as an example project was that it included a large number of common control components, which are necessary when designing a general-purpose HMI. After the XML file output by the sample project was read, it was converted into a Web application. Figure 9(b) displays additional SPA implementation corresponding to the classic GPX components. The control link of GPX in the WINPC32 system was running in the software's real-time mode, in which the SPA had the lowest priority of all the control processes in the controller. Thus, the latency in the data obtained through Internet communication was not considered a serious concern.

2.1. Industrial Controller Case Study. This case study was supported by Taiwan's Ministry of Science and Technology and involved a quasicontinuous wave (QCW) laser-cutting machine, as shown in Figure 10(a), that used a conventional industrial controller. The control parameters of the QCW laser-cutting machine included laser power, pulse width, repetition frequency, and speed of the laser head movement, which required adjustment for different processing and material specifications. Variation in the laser vaporization

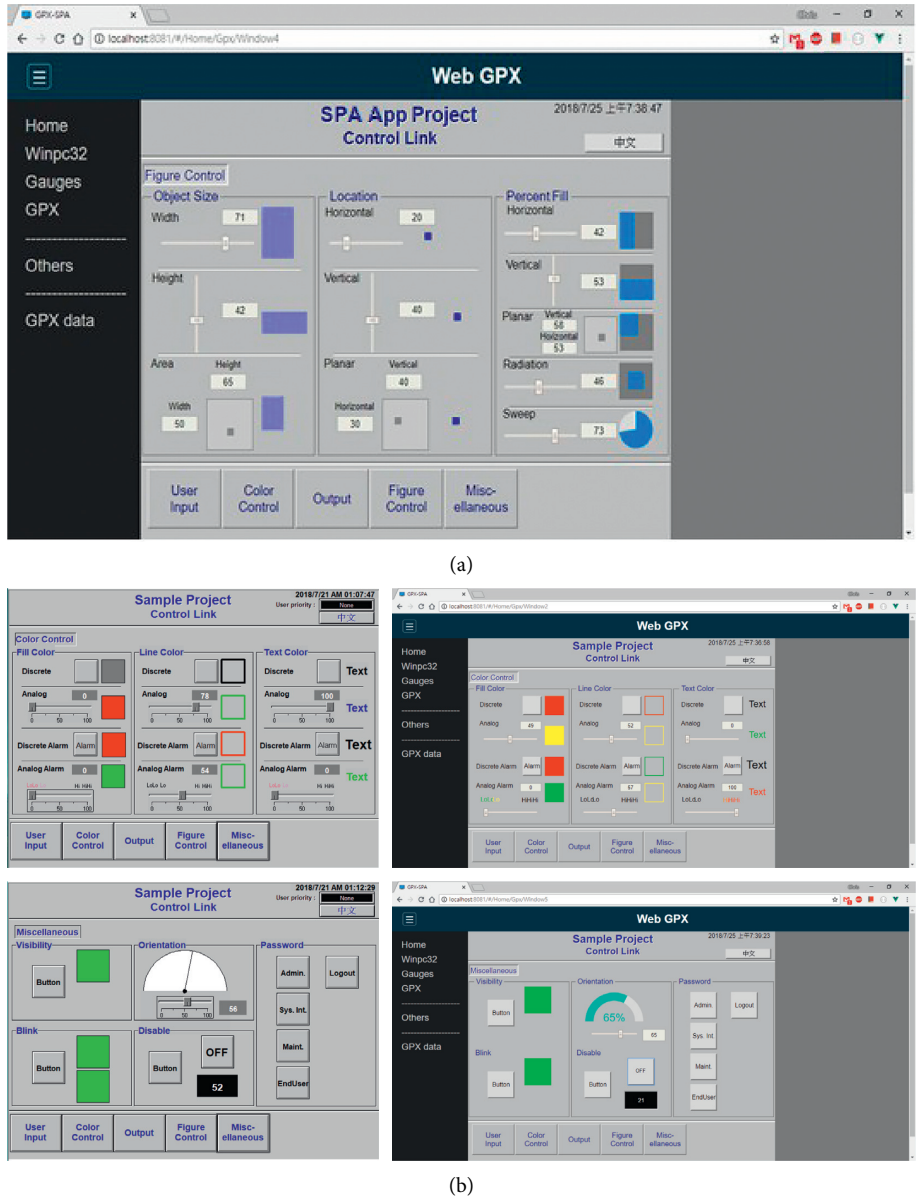
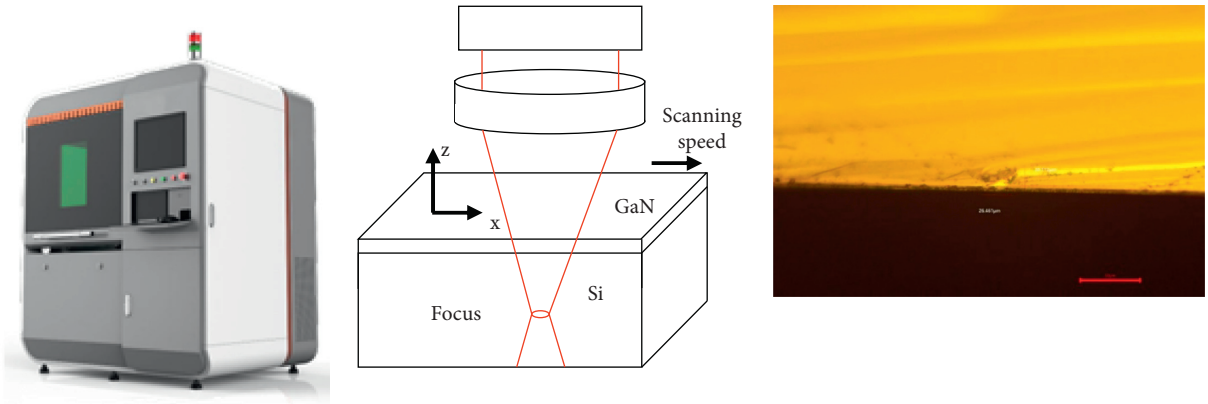


FIGURE 9: Web-based HMI results of the SPA project. (a) SPA example project. (b) Other mapping from GPX (left column) to the SPA page (right column).

zone is a determining factor of the accuracy and quality of laser processing. Therefore, real-time in situ monitoring, which conventional controllers lack, should be added to improve the temperature distribution of laser cutting. The user must also change the control parameters according to the temperature distribution of the remote site.

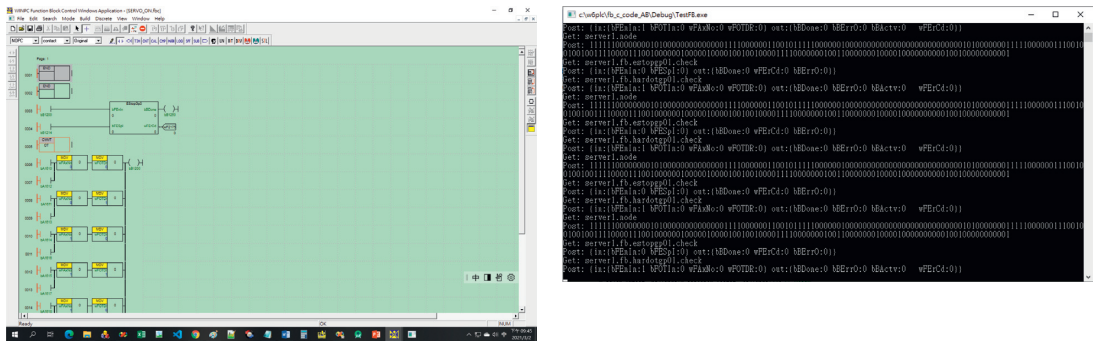
We added an infrared thermal imager to the QCW laser-cutting machine to monitor the material vaporization temperature. The QCW laser-cutting machine had an XY machine table and a Z-axis assembled with an IPG Photonics 1-kW laser head. We used the QCW machine to perform stealth laser dicing of the gallium nitride (GaN) on a silicon wafer, as shown in Figure 10(b). We fixed the value of the numerical aperture to 2.5 cm and changed the focal length of the various focusing lenses. When the focal point was 2.5 cm, as Figure 10(c) shows, the temperature inside the wafer changed drastically. With such a high-value aperture mirror, the laser beam radius had a high focusing effect inside the wafer. At first, the laser energy was small, and the temperature near the focus area rose. As the upper absorption coefficient increased, the laser light energy caused a sharp rise in temperature above the focus, resulting in damage to the material and creating scars. However, throughout processing, the surface temperature of the GaN layer did not substantially rise.

The motion control on the XY table required to displace the wafer from the focus of the laser and the focus length control required on the Z-axis were performed using a hard real-time control system, WINPC32. The software was written in FB in accordance with IEC61131-3 standards.



(a) (b) (c)

Figure 10: Stealth laser dicing using QCW laser. (a) QCW laser-cutting machine. (b) Laser position control. (c) Microscopic images of GaN on silicon control.



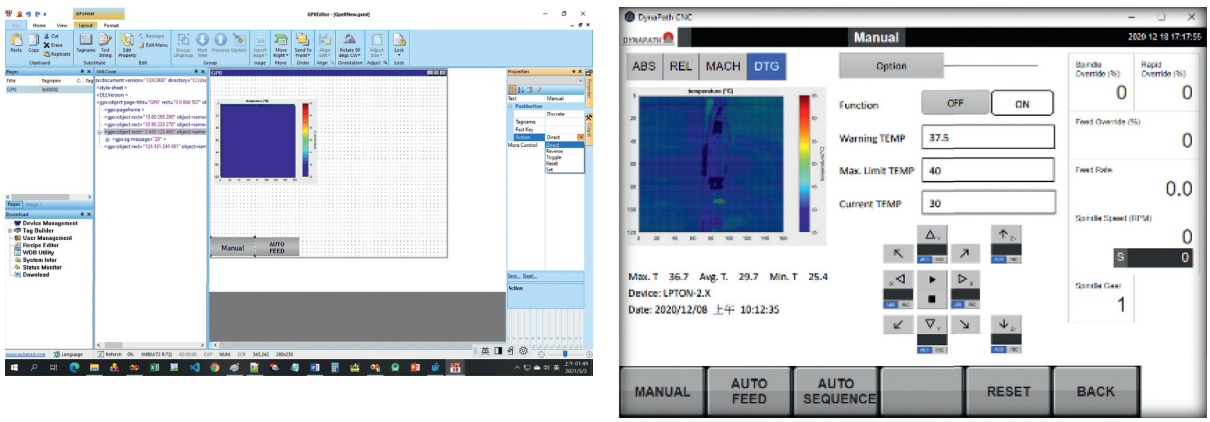
(a) (b)

Figure 11: JASON data streaming from the PLC. (a) FB for logic and motion control. (b) Data subscription and publishing.

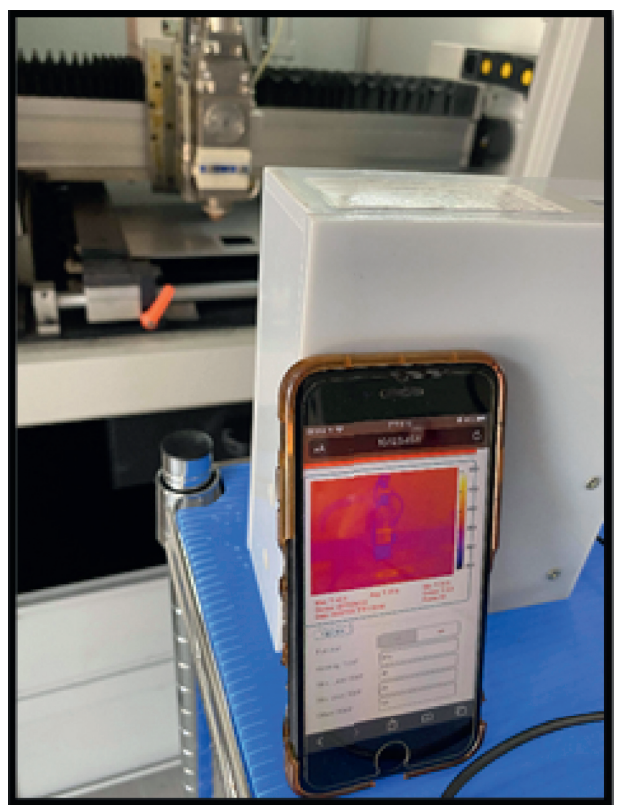
The program flow, as shown in Figure 11(a), is processed between the SPA and the internal PLC controller through the Node.js server. The digital I/O map can be transferred into a series of 0's and 1's. The analog data of the FBs transferred into JSON are shown in Figure 11(b) as an example. The SPA page following the component conversion from the GPX page can utilize the JSON data streaming to simultaneously display the controller and the sensor statuses on smartphones by using the MVC architecture and the client-side rendering, as shown in the flowchart in Figure 5. The PLC was running in a hard real-time environment and using real-time kernels, including RTX from Microsoft. The JSON data streaming publishes the hard real-time control data into software real-time monitoring data that the client SPA page is subscribing.

A new control page corresponding to the material vaporization temperature monitoring with an infrared thermal imager was added to the conventional controller. The execution of the Web-based HMI on the SPA technology for the QCW laser-cutting machine is shown in Figure 10. We first edited the control page on GPX in WINPC32 and then converted it into web page components with the structure displayed in Figure 6. Figure 12(a) shows the process of adding the HMI components one by one to the GPX page. Each component is linked to one of the hard real data defined in the FB, as shown in Figure 12(b). A thermal image was taken at 10 fps and was encoded through the vfw32.lib in a soft real-time process. The receipt download for the laser-cutting control can be implemented in a soft real-time process, as shown in Figure 12(c). The Web-based SPA can be accessed through Node.js.

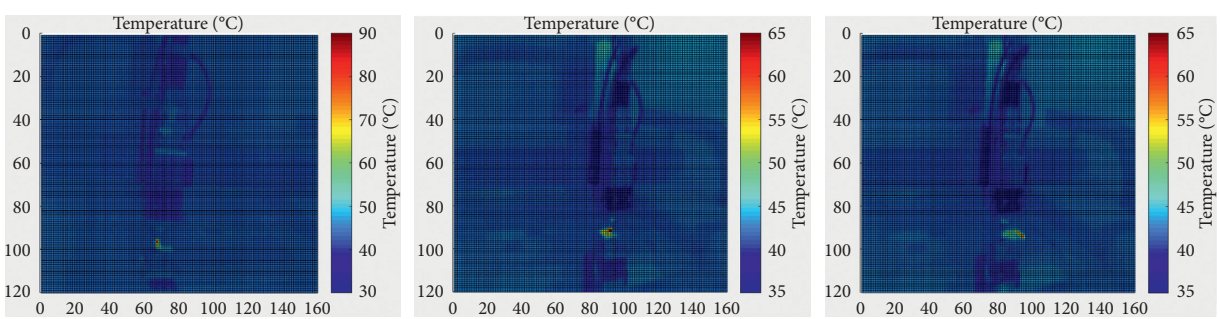
SPAs on smartphones enable users to remotely monitor the laser-cutting process in real time. With SPAs, experts at remote sites can adjust the corresponding control parameters on a web page and monitor the temperature distribution changes of the laser vaporization zone subject to control adjustment in real time. This is a breakthrough for QCW laser machines, which up until now had not allowed remote users to access measurement data and control parameters. The studies reported in [3, 4] entail the transfer of data from the traditional SCADA or PLC to a PC. Our study implemented Web-based technology in PC-based industrial controllers without altering the original Modbus



(a) (b)



(c)



(d)

FIGURE 12: Example of Web-based HMI using SPA on the QCW laser machine. (a) Implementation of GPX. (b) Example of control. (c) SPA on web page. (d) Thermal images of laser dicing.

communication arrangement. Our general-purpose Web-based HMI developed for an industrial controller has the following features:

(1) Direct conversion of the XML file output by the GPX maker to the Web version of the HMI

(2) Use of browsers on different operating systems to process real-time data streaming

(3) Remote real-time monitoring and control for conventional industrial controllers

(4) Simplified program maintenance and automatic adjustment of the web page view on mobile devices

3. Conclusions

Millions of industrial controllers around the world still require IoT technology to transmit controller instant messages across platforms. The first step for software engineers, which may also be the most difficult step, is to find a simple solution to connect conventional controllers to the Internet without rearranging the device's wiring. This study demonstrated a simple strategy for conventional controllers to enter Web-based applications. The solution was to convert the necessary control pages from the conventional controller's HMI into a Web-based HMI. The control pages must be converted into XML at the beginning of the process. XML and JSON enable the information to flow through cross-platform devices across a variety of operating systems and Web browsers. Data flow can be controlled and monitored by using SPAs implemented through AJAX and WebSocket, which can communicate with the back-end Node.js server to transfer data. As reported in the "Results" and "Industrial controller case study" sections of this paper, we converted a number of commonly used control components from conventional industrial controllers to Web-based HMIs. We demonstrated that conventional machines that had not allowed remote users to access measurement data and control parameters could be viewed and controlled through the Internet. The integration of easy-to-access and open-source software such as AJAX and the Node.js server enabled the transformation, making conventional industrial controllers compatible with popular IoT. The results of our research can serve as a reference for software programmers seeking to modernize standalone conventional industrial machines and their controllers. OpenVPN has implemented many features for authentication, encryption, and management that may allow us to use VPN communication in the future to secure network traffic in our system.

Data Availability

The XML format data used to support the results of this study were originally exported by GPX Windows graphics control software. The JSON format data used for web development are generated by GPX Parser and can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Ministry of Science and Technology, Republic of China, under Grant no. MOST 109-2622-E-262-005-CC3. This manuscript was edited by Wallace Academic Editing.

References

[1] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): an analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[2] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, "Industrial internet of things: recent advances, enabling technologies and open challenges," *Computers & Electrical Engineering*, vol. 81, Article ID 106522, 2020.

[3] S. Phuyal, D. Bista, D. Bista, J. Izykowski, and R. Bista, "Design and implementation of cost efficient SCADA system for industrial automation," *International Journal of Engineering and Manufacturing*, vol. 10, no. 2, pp. 15–28, 2020.

[4] Y. Kondratenko, O. Kozlov, O. Gerasin, A. Topalov, and O. Korobko, "Automation of control processes in specialized pyrolysis complexes based on web SCADA systems," in *Proceedings of the IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Bucharest, Romania, September 2017.

[5] I. Qasim, M. W. Anwar, F. Azam, H. Tufail, W. H. Butt, and M. N. Zafar, "A model-driven mobile HMI framework (MMHF) for industrial control systems," *IEEE Access*, vol. 8, pp. 10827–10846, 2020.

[6] G. Caiza, A. Nuñez, C. A. Garcia, and M. V. Garcia, "Human machine interfaces based on open source web-platform and OPC UA," *Procedia Manufacturing*, vol. 42, pp. 307–314, 2020.

[7] R. Sood, G. Singh, and S. K. Chawla, "Single page application: architecture and application," *Recent Trends in Programming Languages*, vol. 6, no. 1, pp. 27–30, 2019.

[8] A. Mesbah and A. Deursen, "Migrating multi-page web applications to single-page AJAX interfaces," in *Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR'07)*, pp. 181–190, Amsterdam, The Netherlands, April 2007.

[9] M. Uehara, "Experiences with a single-page application for learning programming," in *Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 55–66, Yonago, Japan, October 2020.

[10] J. Gunawan and R. R. Kosala, "Genie enterprise resource planning for small medium enterprises implementing single page web application," *E&ES*, vol. 426, no. 1, Article ID 12170, 2020.

[11] https://dojotoolkit.org/reference-guide/1.7/quickstart/introduction/whydojo.html.

[12] F. Shahzad, "Modern and responsive mobile-enabled web applications," *Procedia Computer Science*, vol. 110, pp. 410–415, 2017.

[13] S. Deshmukh, D. Mane, and A. Retawade, "Building a single page application web front-end for E-learning site," in

Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 985–987, Erode, India, March 2019.

[14] P. Chynał and J. Sobecki, "Eyetracking evaluation of different chart types used for web-based system data visualization," in *Proceedings of the IEEE Conference Network Intelligence Conference (ENIC)*, pp. 159–164, Wroclaw, Poland, September 2016.

[15] S. Scepanovic, T. Vujicic, and P. Radunovic, "Web application for lightning activity monitoring system (LAMS)," in *Proceedings of the IEEE Conference Information System and Techologies (CISTI)*, pp. 1–4, Lisbon, Portugal, June 2017.

[16] A. Voinov, C. W. Yang, and V. Vyatkin, "Automatic generation of function block systems implementing HMI for energy distribution automation," in *Proceedings of the IEEE 15th International Conference on Industrial Informatics (INDIN)*, pp. 706–713, Warwick, UK, July 2017.

[17] G. Wang, "Improving data transmission in web applications via the translation between XML and JSON," in *Proceedings of the 2011 Third International Conference on Communications and Mobile Computing*, pp. 182–185, Qingdao, China, April 2011.

[18] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML data interchange formats: a case study," *Caine*, vol. 9, pp. 157–162, 2009.

[19] B. Cvijić, D. Pašalić, D. Bundalo, and Z. Bundalo, "Cloud based web application supporting vehicle toll payment system," in *Proceedings of the IEEE, 5th Mediterranean Conference on Embedded Computing (MECO)*, pp. 489–492, Bar, Montenegro, June 2016.

[20] V. Okanovic, "Web application development with component frameworks," in *Proceedings of the IEEE 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 889–892, Opatija, Croatia, May 2014.

[21] N. G. Obbink, I. Malavolta, G. L. Scoccia, and P. Lago, "An extensible approach for taming the challenges of JavaScript dead code elimination," in *Proceedings of the IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 291–401, Campobasso, Italy, March 2018.

[22] G. Zhang and J. Zhao, "Scenario testing of AngularJS-based single page web applications," in *Proceedings of the International Conference on Web Engineering*, pp. 91–103, Daejeon, Republic of Korea, June 2019.

[23] P. Papcun, E. Kajáti, and J. Koziorek, "Human machine interface in concept of industry 4.0," in *Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pp. 289–296, Kosice, Slovakia, August 2018.

[24] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of industrial internet of things security: requirements and fog computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[25] M. Iqbal and I. Riadi, "Analysis of security virtual private network (VPN) using openVPN," *International Journal of Cyber-Security and Digital Forensics*, vol. 8, no. 1, pp. 58–65, 2019.

[26] Q. Zhang, J. Li, Y. Zhang, H. Wang, and D. Gu, "Oh-Pwn-VPN! security analysis of OpenVPN-based Android apps," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 373–389, Hong Kong, China, December 2017.

[27] A. Skendzic and B. Kovacic, "Open source system OpenVPN in a function of virtual private network," *In IOP Conference Series: Materials Science and Engineering*, vol. 200, no. 1, p. 12065, 2017.

[28] L. Roepert, M. Dahlmanns, I. B. Fink, J. Pennekamp, and M. Henze, "Assessing the security of OPC UA deployments," 2020, https://arxiv.org/abs/2003.12341.

[29] A. Napoleone, M. Macchi, and A. Pozzetti, "A review on the characteristics of cyber-physical systems for the future smart factories," *Journal of Manufacturing Systems*, vol. 54, pp. 305–335, 2020.

[30] http://hacontrols.com.tw/Products/Product-typeA-2.asp?nplSinuYwbaIxJ6ZkoZI.