Remote Controlling and Monitoring of Safety Devices Using Web-Interface Embedded Systems

A. Carrasco, M. D. Hernández, M. C. Romero, F. Sivianes, and J. I. Escudero

Dpto. Tecnología Electrónica, Universidad de Sevilla
Avda. Reina Mercedes S/N, 41012 Sevilla, Spain
{acarrasco,mdhv,mcromero,fsivianes,ignacio}@us.es

Abstract. To date, access control systems have been hardware-based platforms, where software and hardware parts were uncoupled into different systems. The Department of Electronic Technology in the University of Seville, together with ISIS Engineering, have developed an innovative embedded system that provides all needed functions for controlling and monitoring remote access control systems through a built-in web interface. The design provides a monolithic structure, independence from outer systems, easiness in management and maintenance, conformation to the highest standards in security, and straightforward adaptability to applications other than the original one. We have accomplished it by using an extremely reduced Linux kernel and developing web and purpose-specific logic under software technologies with an optimal resource use.

Keywords: Embedded systems, web applications, open source, safety devices, web interface, remote controlling.

1 Introduction

Nowadays, physical access control systems play a key role in corporation's security systems nowadays. They allow intrusion detection and grant access to authenticated users to facilities, devices and other sensible elements located in corporations. Their core is formed by hardware elements such as sensors, and other higher-level devices that allow their management.

The main drawback of this approach lies in the need of updating every single copy of the administration software each time a new update is released. Moreover, there is a strong dependence between the software and the hardware platform it runs on, entailing at worst the replacement of hardware parts to support new software features.

The main goal we proposed when designing the device described within this paper was designing from the scratch, and implementing, a new device that allows monitor and administer an access-control system remotely. This access-control [1] system consisted of multiple hardware parts monitoring unauthorized entrance to rooms within a corporation facility. The system had to surpass the previously commented drawbacks and finally, the device should be easily plugged into any Ethernet network offering its management features through a web-based interface.

Another objective, but not less important, was building a highly secure system.

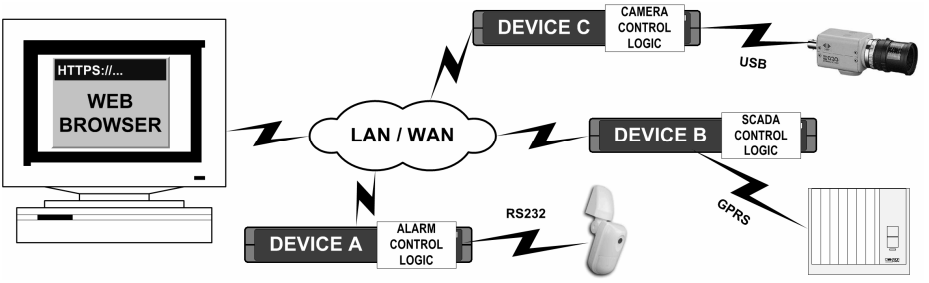

Fig. 1. Using the web-based embedded device for different applications

However, on a long term, the goal was to design a flexible system that could easily be adapted to other uses than monitoring access-control systems, with the least possible modifications to its hardware and software structures.

Apart from the obvious benefits enjoyed by final users, from the developer point of view, having a flexible platform implementing all the hardware parts and interfaces reduces development effort, thus reducing costs.

On the other side, convergence of software and hardware into a single device brings many advantages not only to access-control-orientated systems, but also to other specific-purpose systems such as Supervisory Control and Data Acquisition (SCADA) [3] systems, to give an example.

The solution to all these requirements comes by hands of an embedded platform [2] that implements software logic and hardware interfaces into a single device. The embedded system uses a hardware platform, which together with a modularized design of software components, allow adaptation to new applications.

2 Key Features of the System

Next we will discuss in detail the most noticeable features offered by the embedded device, both to final users and to developers themselves.

2.1 Compatibility

Using a web-based interface provides full software compatibility with any external operating system. Users can operate the device trough a web navigator application, available in all modern operating systems, (using PCs, cell phones…).

On the other hand, having a device based on Ethernet technology with a fully customizable IP makes integrating it into any corporative network a very straightforward process.

2.2 Monolithic Structure

One of the main goals achieved by this platform is to be designed in a way users perceive it like a *black box*. The system, once connected to the hardware it is designed to work with, allows interoperating with the subjacent hardware through the

web-based, built-in administration logic, so that users do not need to know the way devices lying below this platform work.

In a practical case, the box is connected to a Texecom's "Premier" alarm control panel through a serial port, having an Ethernet [4] port so that it is accessible from a Local Area Network and/or Internet. Users, using a web-browser, will connect to a given IP address and receive graphical information about the state of surveyed locations, performing any needed actions unaware of the panel and sensors connected to it.

2.3 Security

Security requirements for parts used in these applications (e.g. access control systems [5]) are very tight, because any security flaw in its software components might compromise the integrity global system.

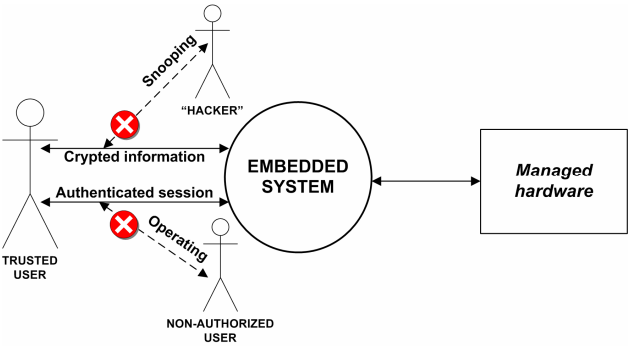

Fig. 2. Securing the embedded system

Therefore, some actions have been taken to ensure that only trusted users can make use of the embedded device's capabilities.

Firstly, information sent and received by our system is secured by means of Secure Socket Layer (SSL) encrypted HTTP connections [6] for all web communications. This means anybody is able to view any sensible information.

Furthermore, user authentication is implemented, which prevents non-authorized users from accessing to the embedded device. All users and groups management is administered from the web-interface; this way, users can be granted to view and/or modify different device's functions depending on the role they play within the corporative organization (e.g. administrators, maintainers, etc…).

Finally, additional securing steps have been taken, as the closure of all unessential and vulnerable network ports (e. g. Telnet [7])[8].

2.4 Ease in Management and Maintenance

Web logic provides a graphical interface that allows any user to use a web navigator to supervise the status of all the devices. It can also be used to deal with all other appropriate system operations such as bypassing of locks, deactivation of alarms, modification of parameters (e.g. the box's IP [9] address), or even updating the

system logic. Being able of updating system software logic from a remote computer is one of the strongest points of this architecture; this introduces many advantages:

- Final users do not need to update or change any software neither any hardware component in their systems, so they can use new features from a first moment.
- Technical assistance and maintenance can be carried out from remote computers, thus physical presence is avoided and maintenance expenses are severely reduced.
- In the case of a system hardware failure, it can be easily replaced by a new one, again without the need of any additional update at users´ environments.
- The option exists of making the system to automatically check for updates (by asking administrators for authorization or by using a fully automatic mode- update itself).

3 System Design

When designing software for embedded systems, we are subject to serious restrictions due to the limited system resources available, such as the processor clock frequency, RAM, and ROM. Other major parameters for the design of the overall system are the processor's power consumption and the cost of the processor. At the same time, there is an increased demand to improve the software-based functionality in the individual device.

Hardware used to build the embedded system consists of all-in-one board having:
- A RISC processor
- 2 MB of flash memory and 8 MB of RAM memory
- Two 100-base-t Ethernet ports, with unique MAC [10] addresses
- 3 serial ports: two RS-232 ports and one USB [11] port

This set-up provides a limited but powerful-enough platform to develop any networked device that can easily adapt to any purpose; its functionality can range from acting as a network firewall to -by simply updating its software logic- be used for communicating with any device via its serial or network ports.

The system can also easily be expanded to add different ports, as e.g. RS-485 or GPRS links [12].

3.1 Operating System

The system is built over a Linux [13] core, which due to the limited hardware resources available, needs to have a very small size. So, we have used some solutions to reduce the size of the embedded Linux operating system:

- All unneeded drivers and services have been removed.
- Only essential shell commands are kept, with reduced functionality. Furthermore,
- they have been put together into a common executable.
- O.S. is also compressed when stored into the flash memory, being decompressed to RAM memory to be effectively executed.

Linux daemons such as HTTP/FTP servers, security suites (firewall, SSL, etc.) or custom device controllers are also deployed as a part of Linux operating system, which are upon booting the system.

3.2 Software Components Design

Due to low spare flash space and RAM memory available, custom software components also need to be small both in size and memory use.

The best approach here is using software components developed in pure C language. Also, server-side scripting languages such as PHP [14] or ASP [15] cannot be used in such an environment, leading us to use client-side JavaScript language and Common Gateway Interface (CGI) [16] executable programs to build the web logic.

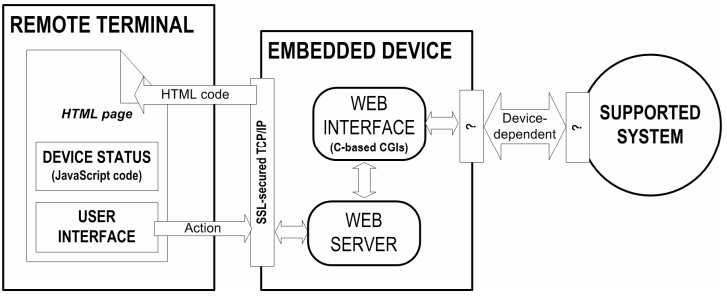

Fig. 3. Web software logic as the interface between the browser and the supported system

Regarding application efficiency, all software components have been developed using a thread-based architecture when concurrent actions (e.g. serving network petitions, etc…) are needed. This improves performance thanks to the parallelization provided by the thread-based approach.

However, special care must be taken with the memory threads use, as they may exhaust the limited memory resources available in the AXIS platform. Memory depletion issues cause erratic behavior not only the consumer applications, but also in other programs and the operating system.

Limiting the number of threads an application can launch is a good mechanism to prevent memory-starvation issues. It may reduce functionality, but it avoids more severe consequences than those software malfunctions can cause over trusted systems; such failures might be exploited by malicious people.

3.3 Real-Life Application: “Indalo”, a “Premier” Alarm Panel Controller

All this theory has been brought into reality in the form of a device implementing a web-based interface that allows administering a “Premier” alarm central device. The alarm central, in its turn, is connected to several alarms distributed along a corporation. This real-life appliance has been successfully deployed into the local network of a national aero-spatial company, allowing security staff to detect unauthorized accesses to its different secured rooms.

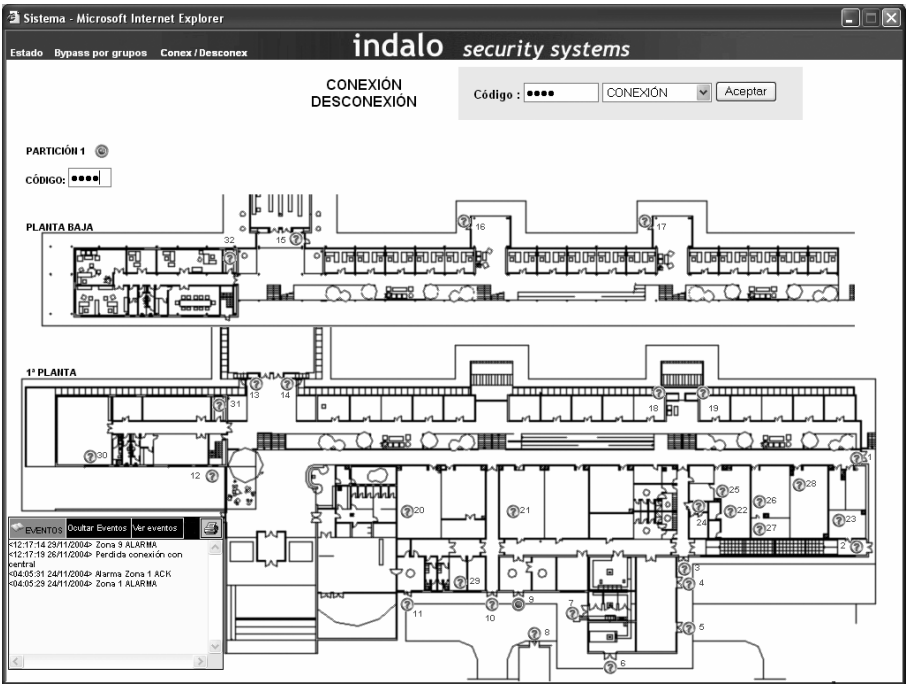

Fig. 4. Screenshot from web interface of "Indalo" access control management device

The "Indalo" device – name of this customization for this general-purpose embedded system - can be accessed through the corporation network, allowing users to have the correct administration rights, supervise and administer the whole access-control system. Both on demand and "real-time" HTML content are provided by the box, being the later periodically updated so it always displays an up-to-date status of the system.

In addition, system's logic provides additional administration web-pages, allowing users to create customized statistical reports based on different policies, such descriptions about non-allowed accesses or other system events that have taken place in a given lapse of time.

As commented above, we are forced to use CGI executables for both transmitting actions taken on web pages to the alarm central and reflecting the status of the different alarms administered by the alarm central.

CGI executables that analyze the status data received from the alarm system are excuted periodically, feeding the HTML pages by creating the JavaScript code so the browser can show the status of the different alarms (bypassed, alarmed, etc.).

Commands to be taken over the alarm system are forwarded from the browser to other CGIs, which analyze the command type and its parameters and send the correct command packet structure to the alarm system via a RS-232 serial connection.

Result of commands operation received from the alarm system is then parsed and JavaScript code is created so HTML pages can reflect the result.

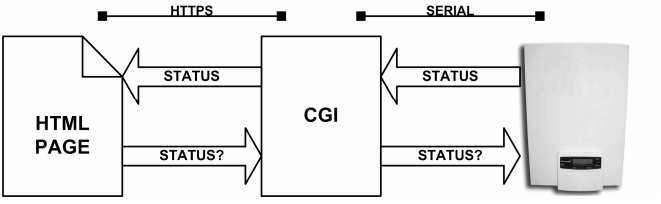

Fig. 5. Embedded system using CGIs to communicate with "Premier" alarm control panel

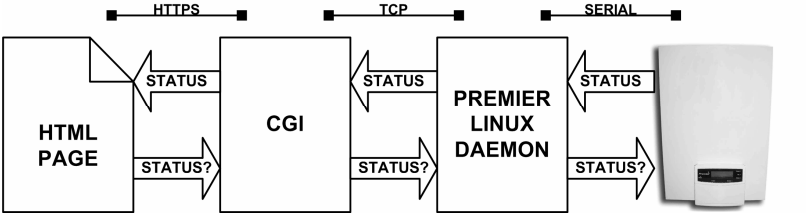

Fig. 6. Separating device-dependent code to be on daemons and HTML-dependent code on CGIs

With the objective of having a more flexible system where CGIs need not be modified when so are low-level components, we have decoupled low-level libraries that communicate with external hardware from CGIs.

Within this new structure, libraries are now located into new executable components designed to work as Linux daemons that serve to requests coming from CGIs using local network sockets. The combined use of sockets and threads allows for multiple CGIs petitions to be simultaneously served.

There is an inherent trouble derived from the way Linux O.S. handles serial ports, which means that concurrent accesses to serial port may cause data sent from different CGIs to be interleaved, and thus making the "Premier" alarm panel receive mixed data it cannot understand.

We have solved this drawback using a shared queue where the daemon inserts all petitions coming from CGIs, so they are served in a serialized manner.

4 Conclusions

The Linux-based embedded system described in this paper provides a flexible and expandable platform that can be used for different purposes.

Once connected to the corporative network, the system can be remotely managed using web connections, so any authorized user located anywhere can make use of its features with a PC, cellular phone or any other device supporting web-surfing.

Security is another strong point in this platform. All information received and sent by it is encrypted using SSL technology. Also, users accessing the device will need to identify themselves; this prevents users from using features other than the ones they are allowed to.

The functionality of this system can be remotely extended, or transformed to support new applications, with just a new firmware (software) update.

This provides a more cost-effective solution for both final users and developers than what was offered by previous systems, where changes in the system meant that the replacement of hardware parts, or displacement of technical staff was necessary.

In a world where companies are increasingly giving services to international customers, the capability of providing distant support makes all the difference.

The development of this kind of system shows that –in spite of the restrictions in memory and CPU resources- it is possible to build a powerful general-purpose embedded device, which can be programmed to serve a specific purpose.

This goal has been achieved thanks to a design where efficient software technologies and techniques, as well as common hardware interfaces, are the keys to success.

Acknowledgements. The work described in this paper has been funded by the Ministerio de Ciencia y Tecnología within the I+D+I National Program through the project with reference number TEC2006-08430.

We´d also like to thank ISIS Engineering (Seville) for providing us with prototypes, and Medina-Garvey electrical company for letting us use their facilities.

References

1. Konicek, J., Little, K.: Security, ID Systems and Locks = The Book on Electronic Access Control. Butterworth-Heinemann (1997)
2. Yaghmour, K.: Building Embedded Linux Systems. O'Reilly, Sebastopol (2003)
3. Boyer, S.A.: SCADA = Supervisory Control And Data Acquisition, 2nd edn. SA – The Instrumentations, Systems and Automatic Society, New York (1999)
4. Information technology – Local area networks – Part 3: Carrier sense multiple access with collision detection. IEEE 802.3 (1993)
5. Ibarra-Manzano, M.A., Almaza-Ojeda, D.L., Aviles-Ferrera, J.J., Avina-Cervantes, J.G.: Access Control System Using an Embedded System and Radio Frquency Identification Technology. In: IEEE, Electronics, Robotics and Automotive Mechanics Conference – CERMA 2008, pp. 127–132. IEEE Press, Los Alamitos (2008)
6. Rescorla, E.: HTTPS = HTTP over TLS. IETF RFC 2818 (2000)
7. Postel, J., Reynolds, J: Telnet protocol specification. IETF RFC 854 (1983)
8. Yan-ling, X., Wei, P., Xin-guo, Z.: Design and implementation of secure embedded systems based on Trustzone. In: International Conference on Embedded Software and Systems – ICESS 2008, Sichuan, pp. 136–141. IEEE Press, Los Alamitos (2008)
9. University of Southern California: IP = Internet Protocol. IETF RFC 791 (1981)
10. Leon-García, A., Widjaja, I.: Communication Networks, 2nd edn. McGraw-Hill, New York (2003)
11. Technical guide of USB 2.0. USB Implementers Forum (2001), `http://www.usb.org`
12. GPRS – Service Description; Stage 2. ETSI GSM 03.60 (2000)
13. Siever, E., Figgins, S., Weber, A.: Linux in a Nutshell, 4th edn. O'Reilly, Sebastopol (2003)
14. Hypertext Processor Scripting Language – 5.0.2. The PHP Group (2004), `http://www.php.net`
15. Mitchell, S.: Designing Active Server Pages. O'Reilly, Sebastopol (2000)
16. Robinson, D., Coar, K.: CGI = The WWW Common Gateway Interface Version 1.1. IETF RFC 3875 (2004)