Design of a Smart Lock on the Galileo Board

Matias Presso, Diego Scafati, José Marone
Fac. de Ciencias Exactas, Univ. Nacional del Centro de la Provincia de Buenos Aires, Argentina
marone@exa.unicen.edu.ar

Elías Todorovich
Univ. Nacional del Centro de la Pcia. de Bs. As. and Universidad FASTA, Argentina
etodorov@exa.unicen.edu.ar

***Abstract*—In this work, a WiFi enabled door lock allows users to lock or unlock doors by codes that are managed from a computer or phone where new codes are issued or deleted including temporary codes to guests or office personnel. This work demonstrates an efficient smart lock system that can be configured remotely using Intel Galileo boards. The on-board server is connected through a mini-PCIe Wi-Fi card supported by Galileo. The Web server provides real-time information to remote administration regarding interesting events on the smart lock in the form of a responsive webpage. In addition, all the source code of the project is open and available for developers.**

***Keywords*—smart lock; open source; Intel Galileo; IoT**

I. Introduction

Internet has evolved from connecting computers to connecting IP-enabled things that can be employed for real time remote monitoring in the Internet of Things (IoT).

Between 2013 and 2014, Intel donated 50,000 Galileo boards to about 1,000 universities worldwide. This development board is compatible with Arduino to enable university students to innovate in the IoT. One of the impacts of this large-scale donation is a number of papers published with designs and applications related to IoT. For example, [1] demonstrates a method of real time remote monitoring of environmental parameters using Intel Galileo. In this method, an on-board server is connected to the Wi-Fi router that acts as a gateway for this network. The Web server provides the real time sensor information with dynamic updates in the form of a webpage to the remote client. Another application is a fall detection system for the elderly [2].

Connected devices and the related IoT technologies generate various smart embedded products that can be used in diverse applications, like home automation systems. A smart lock is one of the most important parts of a smart home. Authors in [3] present a smart lock system focusing on security issues. In a relatively older work [4], a ZigBee module is embedded in the digital door lock and the door lock acts as a central main controller of the overall home automation system.

The goal in this open source project is to develop an efficient smart-lock system using essentially a Galileo board.

A. Intel Galileo

Intel Galileo is the first in a line of Arduino-certified development boards based on Intel x86 architecture and is designed for the maker and education communities [5]. This project uses the first of the two Galileos, referred to as Gen 1, which was released in October 2013. Galileo runs a Linux operating system with the Arduino software libraries, so that existing "sketches" run on Galileo’s processor. The board is also designed to be hardware and software compatible with the Arduino shield ecosystem (See Fig. 1). It can also be programmed by the Arduino software development environment (IDE) and libraries.

The 32-bit Quark SoC X1000 chip in Galileo is an attempt by Intel to compete within the IoT market. It is a single core, single thread processor based on an x86 Pentium design, and it runs at a clock speed of 400MHz. It has 512KB of embedded RAM and a 16 KBytes on-die L1 cache.

Galileo storage includes a programmable 8MB NOR flash, whose main purpose is to store the firmware (or bootloader) and the latest sketch, and a 256MB DRAM. Other key features include a micro-SD connector slot for cards of up to 32GByte of storage, a 10/100Mbps (bits per second) Ethernet port, a mini PCI-Express slot, a USB 2.0 host connector, and an RS-232 serial port connected via a 3.5mm jack. The USB Device ports allow for serial (CDC) communications over USB. This provides a serial connection to the Serial Monitor in the Arduino IDE.

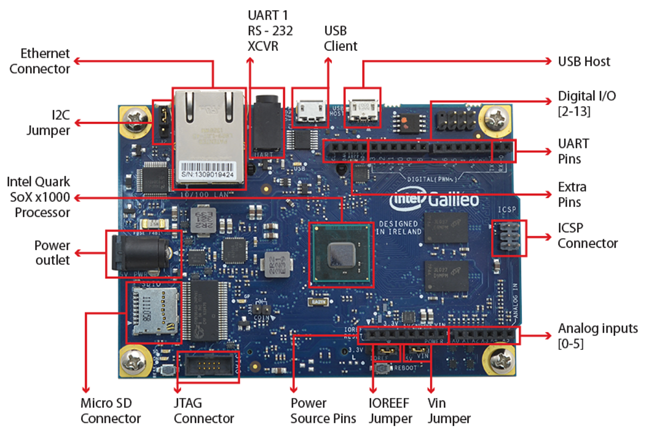


Fig.1. Intel Galileo board front

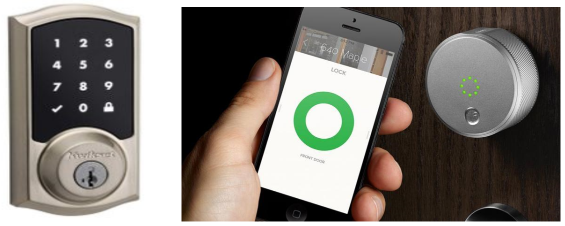


Fig. 2. Smart locks.

II. Smart Lock

A smart or internet-controlled lock is useful for rental properties, home, apartment complexes, fraternities, or office use. This is a WiFi enabled door lock that allows users to lock or unlock doors remotely by a code, know when people unlock the door, and receive text alerts when codes are used. Door lock is managed from a computer or phone where new codes are issued or deleted including temporary codes to guests or office personnel. Some models include a keypad to enter a code and others simply let you use your phone to open and close doors (See Fig. 2).

One of the problems related to remotely managing property access is key exchange. Although it has associated copying and personnel costs, it can cause the disruption of early check-ins and overstays. In addition, there are security risks of unauthorized access. To solve this problem some internet-controlled lock companies even offer online integration with apps from hospitality service providers such as Airbnb.

Small business smart-locks using WiFi allow owners to remotely monitor and control access, to view history logs of employee access, and even to make sure doors are locked. These locks are useful for office suites, health facilities, space sharing offices, interior offices, and utility rooms.

Many smart locks offer a mobile app that allows users to lock and unlock doors with a simple icon tap. Some offer a Web app that lets you control things from your desktop or laptop PC. These apps let you add permanent and temporary users and set access schedules for specific days and times.

The latest smart locks offer features like voice activation, auto-locking features, keyless touchpads for those times when you don't have your phone or your keys, and tamper and forced entry alarms that warn you of a possible break-in, and push, text, and email notifications that let you know who is coming and going in real-time. Some locks also integrate with other connected home devices. For example, you can have your doors unlocked ‘when a smoke or CO alarm is triggered, or have certain smart lights turned on when a door is unlocked. You can even pair your lock with a connected doorbell cam, so you can see who is at the door before you unlock it.

Although there are several smart locks in the market, there are still two obvious reasons to develop an open source solution. One is the price, and the other is that those products have proprietary software and hardware.

III. Development

Readers can download and study the source code and a step-by-step guide (in Spanish) [6] to implement this system in a Galileo board. This section explains the architecture and design of the system, to help readers not only to understand how the project works but also to be able to extend or modify this open source system.

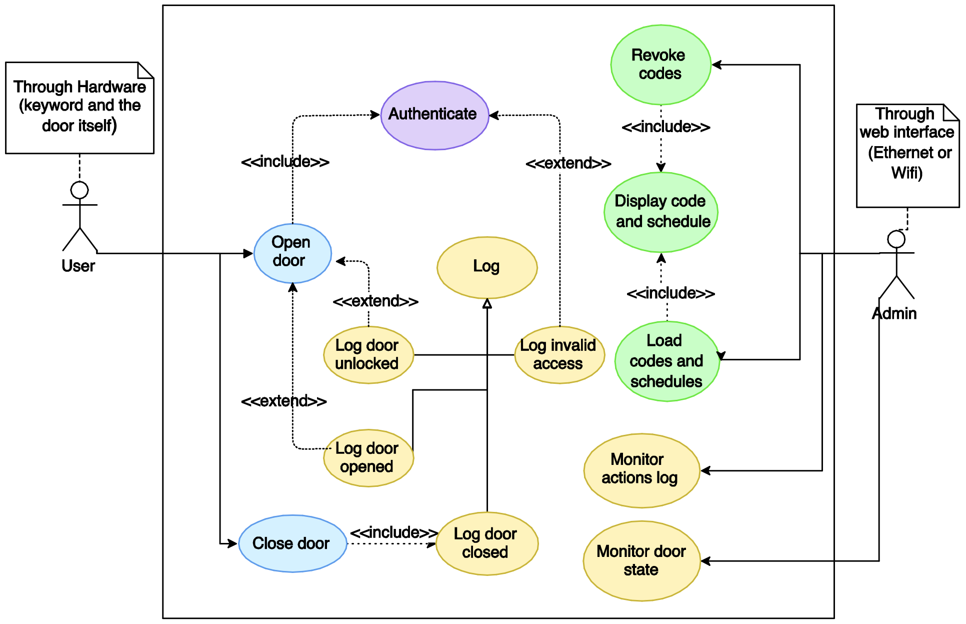


Fig. 3. UML use case diagram with the specification of the smart lock functionality.

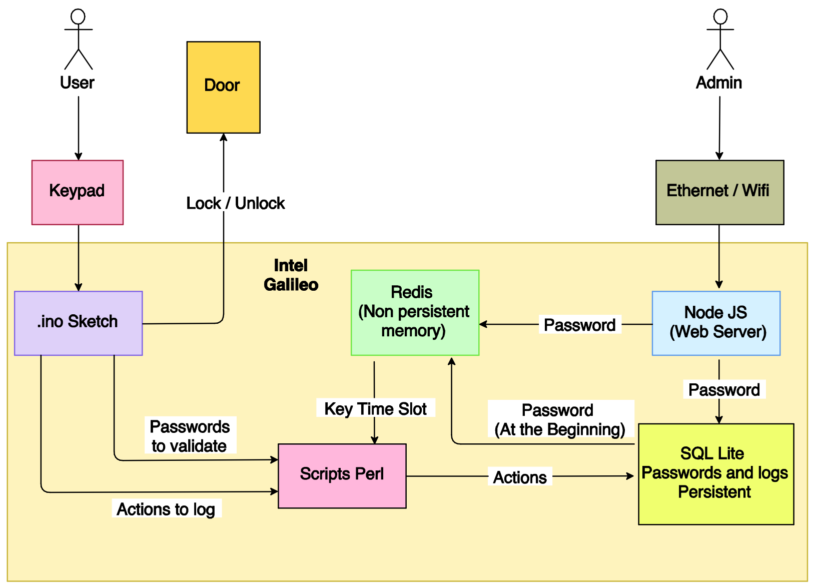


Fig. 4. Smart lock architecture.

A. *Specification*

For the smart lock defined for this project, there are two types of actors: on the one hand, the "users" who will use the lock to open the door and access the restricted site; on the other hand, the "administrators" who configure virtual keys and access date and times, and monitor the operation (see when and who open or close the door, etc.).

Users receive a code or virtual key by some text messaging service. Then, they can enter this code to unlock the door. Administrators can add codes and associate them to users and a time period, delete or disable codes, and see the logs. Logs show a list of users and timeststamps when the door was unlocked, when it was opened, closed or locked, and the time when an invalid code was attempted. Fig. 3 shows the smart lock functionality in an UML use case diagram.

Two additional functionalities were added, a display of the the current door status, and the possibility of synchronizing the internal clock in the smart lock by means of the web interface.

B. *Architecture and Design*

In this project, the Linux operating system (OS) running on the Galileo boards is used several times. This is one of the most important features of these cards. The OS can be accessed from Arduino sketches through system calls, i.e., the system() method, for example, to set up the WiFi card:

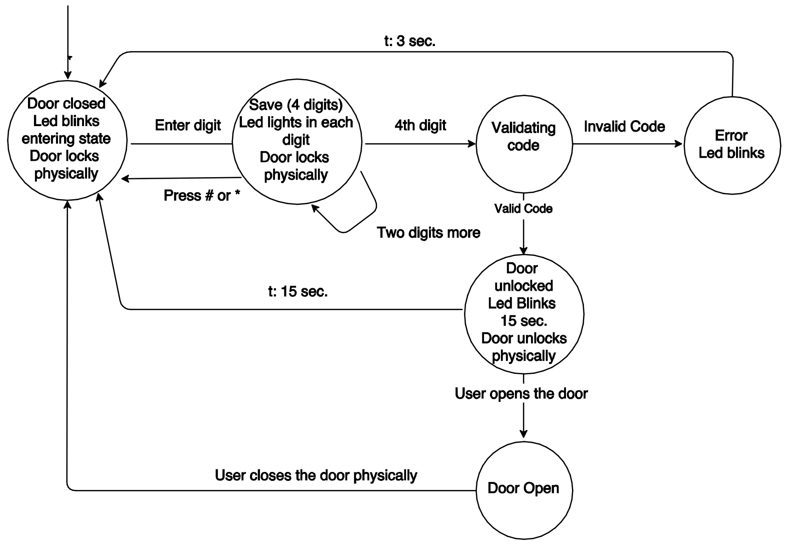

t: 3 sec.
Door closed Led blinks entering state Door locks physically
Enter digit
Save (4 digits) Led lights in each digit Door locks physically
4th digit
Validating code
Invalid Code
Error Led blinks
Press # or *
Two digits more
Valid Code
t: 15 sec.
Door unlocked Led Blinks 15 sec. Door unlocks physically
User opens the door
User closes the door physically
Door Open

Fig. 5. Smart lock FSM.

```
system("connmanctl enable wifi");
```

Although security is one of the biggest IoT challenges, it is not a specific design issue considered in this article. However, as the system is running on a Linux OS, we encourage designers to apply at least the basic security measures such as changing root password, regularly updating packages and firmware, adding a domain name and SSL certificate to their board, among others.

Two forms of storage are observed in Fig. 4. SQLite is used for storing the keys and schedules in a persistent way. SQLite databases are stored as files in the filesystem, making it available after restarting the hardware (persistence). The problem with this kind of storage is the low performance retrieving results, in terms of response time. To improve the overall performance and speed up the response time to user operations, we decided to use Redis, an in-memory storage system. Redis is used to keep the keys in memory so that accesses can be validated quickly, delegating SQLite to the task of keeping the persistence of the data and working in an asynchronous way (without delaying the response time when possible). Other advantages of using Redis are its simple usage (no SQL queries are required) and its integration within all the technologies involved in the project, i.e, a variable stored in Redis can be read from a Perl script, a JS script, or the main .ino script. In other words, Redis is also used as a centralized repository for storing and sharing variables between programs in the OS. The SQLite database contains only two tables, one to store the codes and another to keep a record of the performed actions. Fig. 4 also illustrates the Arduino code in execution as well as a web server in NodeJs (javascript). Intermediate scripts in Perl are used to access persistence options from Arduino (Redis / SQLite).

Fig. 5 shows the finite state machine (FSM) implemented as a sketch in Arduino and running on the Galileo board. When a user enters a digit using the keypad, the FSM goes from *locked* to *writing-key* state. After the user enters a 4-digit key, the FSM goes to the *validating* state, and then to *error* or *unlocked* state if the code is valid compared with the codes in the database. Finally, from the *unlocked* state, the FSM goes to *open* state when the user opens the door. Main state transitions are registered in a log.

C. *Implementation*

This system requires OS services that are not provided by the Linux embedded in the Galileo Board. Fortunately, Intel made available and maintains different Board Images based on the Yocto project (Linux) for extending the capabilities of the board. These custom distributions include drivers and an environment which can contain Redis, NodeJS, Perl, and/or other pre-installed tools and code interpreters, making them an excellent option for projects like this. For our particular project, we used the "Intel® IoT Developer Kit v1.5 Intel® Galileo Board Image".

A mini-PCIe WiFi card Atheros ar5b225 is used. This card is reused; it comes from a laptop that is out of service. However, the Intel® Centrino® Wireless-N 135 and Intel® Centrino® Advanced-N 6205 would be better options as they work with any Linux image provided by Intel.

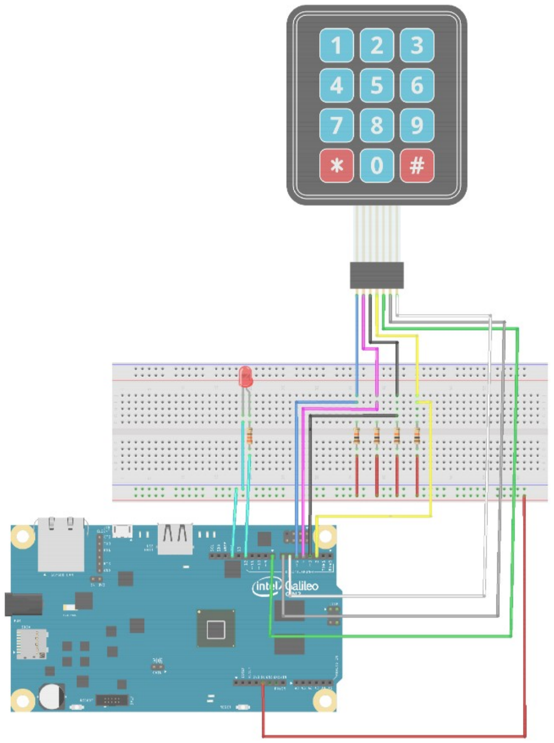


Fig. 6. Keypad connection

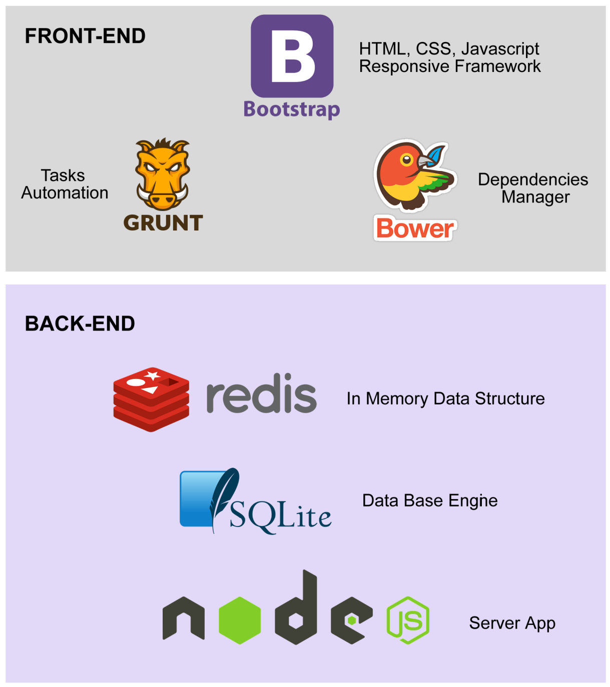
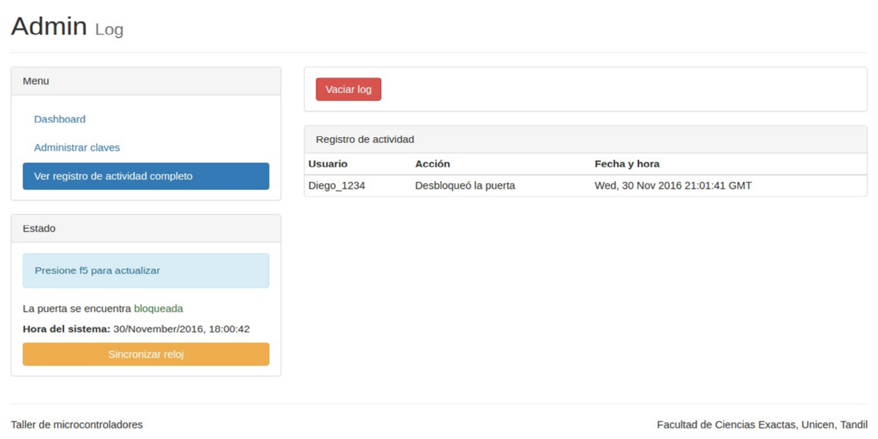


Fig. 7. Front-End / Back-End Tools and Technologies Ecosystem



Registro de actividad

Usuario	Acción	Fecha y hora
Diego_1234	Desbloqueó la puerta	Wed, 30 Nov 2016 21:01:41 GMT

Presione f5 para actualizar

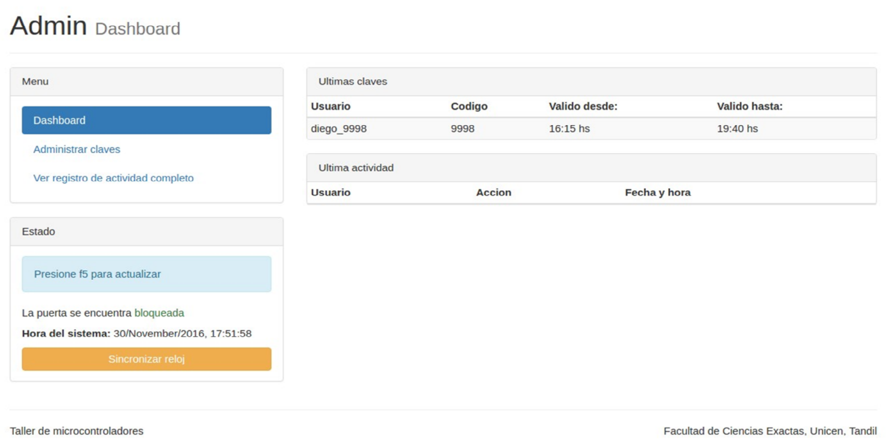
La puerta se encuentra bloqueada

Hora del sistema: 30/November/2016, 18:00:42

Taller de microcontroladores

Facultad de Ciencias Exactas, Unicen, Tandil

Fig. 8. Admin Log Screenshot



Ultimas claves

Usuario	Codigo	Valido desde:	Valido hasta:
diego_9998	9998	16:15 hs	19:40 hs

Ultima actividad

Usuario	Accion	Fecha y hora

Presione f5 para actualizar

La puerta se encuentra bloqueada

Hora del sistema: 30/November/2016, 17:51:58

Taller de microcontroladores

Facultad de Ciencias Exactas, Unicen, Tandil

Fig. 9. Admin Log Dashboard

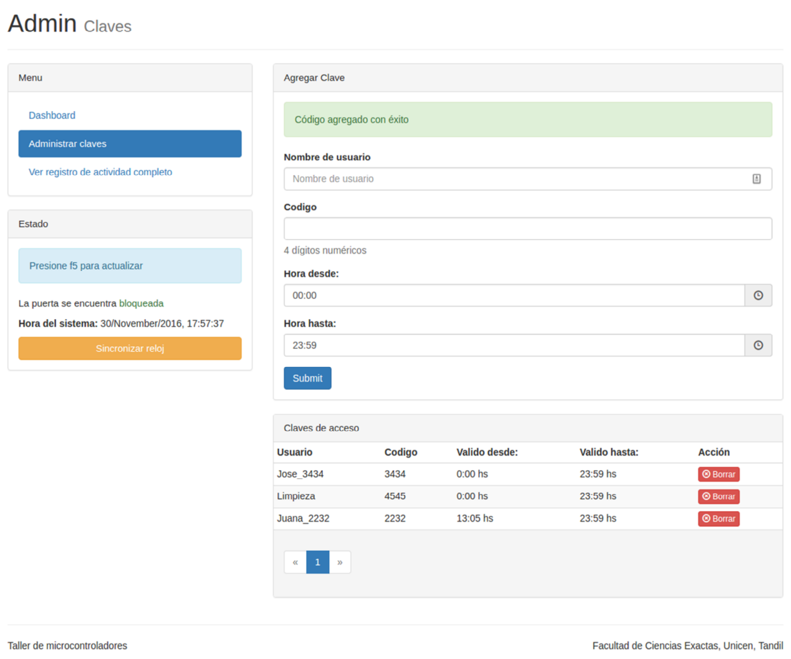
On Arduino cards, the connection from the keypad outputs to the card is direct because the Arduino pins provide internal pull-up resistors, but in this project pull-up resistors were added to avoid glitches, as shown in Fig. 6.

Regarding Arduino programming, the sketch mainly consists of the implementation of the FSM shown in Fig. 5. The FSM library for Arduino is used to implement the FSM states and transitions between states in an organized way. All these states expect an *enter*, an *update,* and an *exit* callback function.

Fig. 7 summarizes front-end and back-end tools and technologies used for the project development. NodeJs [8] is used and configured as a lightweight server application. Due to its single-threaded architecture, it minimizes memory usage and avoids the cost of context-switching between threads. These reasons make it an ideal tool for real-time tasks and GPIO (general purpose input and output) interaction. Bower [9] and Grunt [10] are used as front-end development tools. The former for frameworks, libraries, assets, utilities, and dependencies management, and the latter for tasks automation. Both tools allow a modular development, and ease updating, maintenance and application debug. Bootstrap [11] was included as responsive front-end framework and SQLite [12] as database engine.

Redis [13] is an efficient in-memory key-value database open-source software. Many languages have bindings to Redis such as the ones used in this project JavaScript (Node.js) and Perl.

Figs. 8, 9 and 10 show the web application and database queries for Admin Log, Admin Dashboard, and Password Administration, respectively.



Código agregado con éxito

Presione f5 para actualizar

La puerta se encuentra bloqueada

Hora del sistema: 30/November/2016, 17:57:37

Nombre de usuario

Nombre de usuario

Codigo

4 dígitos numéricos

Hora desde: 00:00

Hora hasta: 23:59

Claves de acceso

Usuario	Codigo	Valido desde:	Valido hasta:	Acción
Jose_3434	3434	0:00 hs	23:59 hs	Borrar
Limpieza	4545	0:00 hs	23:59 hs	Borrar
Juana_2232	2232	13:05 hs	23:59 hs	Borrar

Taller de microcontroladores

Facultad de Ciencias Exactas, Unicen, Tandil

Fig. 10. Pass Admin Screenshot

IV. Conclusion

A complete smart lock open-source system is implemented on a Galileo gen 1 development board. This means that system administrators can access the web server running on the board itself, and the users can enter codes that are validated by on-board software too.

Authors provide a step-by-step guide to install and test the system. Besides, developers around the world can extend and modify the available source code with minimum complexity. Although this is the first available version of the proposed system, its architecture enables modular development, management, automation, and eases updating and maintenance by means of stable, professional, and widely accepted software tools.

Acknowledgment

Authors would like to thank: Intel Corporation for their support by providing Intel Galileo boards under Intel – University donation program as part of Intel higher education to the UNICEN, Argentina, and FASTA University, Argentina, and CIC PBA where M.P. belong to the Professional Staff.

References

[1] J.J. Livingston, and A. Umamakeswari, "Internet of Things Application using IP-enabled Sensor Node and Web Server," Indian Journal of Science and Technology [Online], 8.S9 (2015), pp. 207-212.

[2] G. E. De Luca, E. A. Carnuccio, G. G. Garcia and S. Barillaro, "IoT fall detection system for the elderly using Intel Galileo development boards generation I," 2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI), Buenos Aires, 2016, pp. 1-6.

[3] A. Kassem, S. E. Murr, G. Jamous, E. Saad and M. Geagea, "A smart lock system using Wi-Fi security," 2016 3rd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Beirut, 2016, pp. 222-225.

[4] Y. T. Park, P. Sthapit and J. Y. Pyun, "Smart digital door lock for the home automation," TENCON 2009 - 2009 IEEE Region 10 Conference, Singapore, 2009, pp. 1-6.

[5] Miguel de Sousa, Internet of Things with Intel Galileo, Packt Publishing, 2015.

[6] Diego Scafati, "Guía paso a paso: Cerradura digital con código en Galileo", available in: https://github.com/dscafati/arduino-electronic-door-lock

[7] Scott Rifenbark, "Yocto proyect mega-manual", available in: http://www.yoctoproject.org/docs/2.2.1/mega-manual/mega-manual.html

[8] Galileo Tutorial Networking and node.js Senzations 2014, Jason Wright, available in:

http://senzations.net/wp-content/uploads/2014/66/Senzations14-Networking-nodejs.pdf

[9] Bower Package for the Web, https://bower.io/docs/config/

[10] Grunt Javascript Task Runner, https://gruntjs.com/configuring-tasks

[11] Bootstrap Responsive Framework, http://getbootstrap.com/getting-started/

[12] SQLite Documentation, https://www.sqlite.org/docs.html

[13] Redis Documentation, https://redis.io/documentation