Internet-enabled Access Control System using a Mobile Application

Alexandru Agape
Faculty of Automatic Control and Computer Engineering
"Gheorghe Asachi" Technical University of Iasi
Iasi, Romania
agape.alexandru@ac.tuiasi.ro

Mihai Postolache
Faculty of Automatic Control and Computer Engineering
"Gheorghe Asachi" Technical University of Iasi
Iasi, Romania
mpostol@ac.tuiasi.ro

***Abstract*— The need to secure property or restrict entrance in special areas by controlling access to them has always been extremely important for life safety and personal assets, as well as those of the organizations. Different mechanical or electronic door locks are available today using keypads and passwords, RFID devices, biometric data, in order to prevent unauthorized access in homes, research labs, banks, stores, etc. This paper presents a secure access control system based on a Raspberry Pi3 model B board connected to Internet via the local area network. The system monitors in real-time the front of the door using a camera and face recognition features, and notify the system administrator if anyone spends time in front of the door more than a specified limit. Registered users have allowed access and may unlock the door through a mobile application, as the identification may be done by face recognition, password, or fingerprint, subject of availability. The system administrator also has access to the camera captured frames and may unlock the door remotely per request. A web server running on the Raspberry Pi3 embedded system maintain a local database containing the registered user list and related data, as well as a log file that provide the system administrator detailed information about any detected and recorded events.**

Keywords*— *embedded systems, secure access control, web server, mobile application, Internet of Things (IoT)

I. Introduction

With the new advances in computing and communication techniques, many applications that previously required embedded systems running in a rather isolated manner can now be interconnected in an Internet of Things (IoT) world providing the end user a more integrated view of the whole system and new ways to interact with the environment.

IoT has enabled a transition from smart devices to smart homes, towards smart organizations and smart cities, while new challenges and threats are to be answered and faced. People have to be part of this evolution towards a hybrid man-machine world and for this reason new human-machine interfaces (HMIs) and communication techniques have to be designed and implemented in order to allow a smooth and fair interaction.

The access control is such a typical interaction between a security system and people. Its purpose is to detect and recognize the presence of an individual, uniquely identify it using one or more authentication techniques, log the event in a database and authorize the access [1].

The detection phase is used if the sensors are able to detect the physical presence of people nearby the access control point. It is the case of various types of proximity sensors or camera. In the authentication stage, biometric or non-biometric data are acquired and specific algorithms are used to validate and uniquely identify a person based on a priori stored information: passwords, pictures, RFID, fingerprints or answers to security questions.

The validation phase can be (*i*) fully automated, when the access is granted or refused as a result of an unassisted algorithm, or (*ii*) this phase can be human assisted when the result of the authentication together with relevant data are sent to the system administrator, and finally he may or may not authorize the access with or without new requirements from the solicitant. All the events regarding the access control may also be recorded in a log file for off-line or on-line access, and the system administrator may choose what type of event should instantly be notified of via e-mail or SMS.

II. Related Work

In [1] a Raspberry Pi Single Board Computer with WiFi connectivity was used as an IoT based security, active surveillance system inside a building. The main focus is on intruder and fire detection, being able for live video streaming when needed.

The system proposed in [2] also uses a Raspberry Pi having a biometric scanner for fingerprint directly connected to it together with vibration, fire, temperature, and gas sensors, in an extended smart home automation system. If the fingerprint test is passed, the user needs to login to the web server, introduce a password and answer to a security question.

A hybrid access control security system integrating an access control system and a security system was presented in [3]. It used an XScale PXA270 embedded system running a web server and interfacing to a surveillance camera and sensors for home security.

Until recently, implementation of complex face detection and recognition algorithms on embedded hardware was difficult, and even now some limitations applies when it comes to feed such algorithms in low cost embedded devices. A survey of the main research work on implementing the facial recognition in embedded systems or in special hardware using Field Programmable Gate Array (FPGA) systems for real-time applications as is the case with the access control is done in [4].

Other work [5-10] propose several implementations of door access control systems which are more or less connected to different home security systems and to Internet, directly or via Wireless Personal Area Networks (WPANs) such as ZigBee [5], Bluetooth Low Energy (BLE) [6], or BodyCom [7].

Our proposed secure access control system makes use of client-server custom application and a smart device that people already carry on with everywhere – a smartphone. It is assumed that there is a WiFi network access point available at home, which should be as secured as the smartphone used to request the access control. In the following sections the overall architecture and the features of the proposed solution will be described, as well as the results and the future extensions to be implemented.

III. System Architecture

The secure access control system (Fig. 1) is entirely built around a mini-PC Raspberry Pi3 model B, and includes a Microsoft Lifecam HD 3000 USB 1280x720 webcam, and a circuit for driving a 12Vdc electromagnetic (EM) door lock.

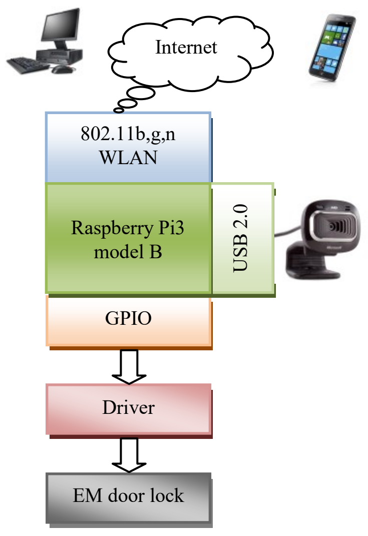


Fig. 1. Overall secure access control system architecture

Raspberry Pi is a series of small Single Board Computers (SBCs) developed by the Raspberry Pi Foundation with the support of the University of Cambridge Computer Laboratory and Broadcom [11].

Raspberry Pi3 model B is the third generation Raspberry Pi empowered by a Broadcom BCM2387 ARM Cortex-A53 Quad Core Processor running at 1.2GHz. It has 1 GB RAM, an integrated GPU VideoCore IV at 400MHz, 10/100 BaseT Ethernet and 802.11b,g,n Wireless LAN, Bluetooth 4.1, 4 USB2.0 ports for standard peripherals, HDMI support, as well as 2 special ports for connecting Raspberry Pi peripherals such as camera and touchscreen. Also, the board has an SD card port for boot loading the operating system and for data storage, as well as 40 General Purpose Input Output (GPIO) pins for interfacing to external sensors and actuators.

Raspbian is a free operation system based on the Debian Linux distribution which is optimized for the Raspberry Pi hardware [11]. It includes over 35,000 pre-compiled software packages bundled for easy installation on Raspberry Pi. The combination of free software and increased power computing, advanced communications, HMI and process interfaces at low cost and small form factor makes of Raspberry Pi3 a good candidate for IoT projects.

IV. The Embedded Controller Custom Application

Due to the real time requirements of the application, and the availability of the quad-core processor of Raspberry Pi3, a multi-threading controller application was designed in C/C++ using the thread class of the C++ library.

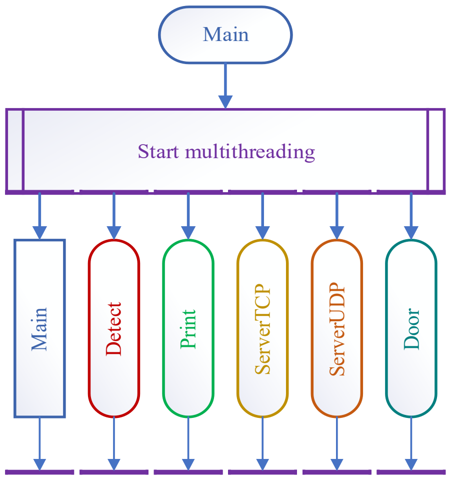


Fig. 2. Multithreading architecture of server application

A. *The Multithreading Architecture*

There are 6 types of threads running in parallel (Fig.2), synchronizing and exchanging information via message queues, as described below:

- The *Main* thread is responsible for application and board initialization, creating the other threads and capturing images from the camera, which will be added in a queue (Q_1).
- *Detect* thread will take a queue image from Q_1, detect faces, if any, put them in a border and add them to another queue (Q_2).
- The *Print* thread picks up images from the queues Q_1 and Q_2 (with higher priority for Q_2) and displays them via the HDMI port.
- The *ServerTCP* thread handles the communication with the mobile application client side. It will receive

requests, process them and answer to the client accordingly, update the configuration XML and add events to the XML log.

- The *ServerUDP* thread send images to registered users if they are allowed and when they want to track what's going on in front of the door.
- The *door* thread is responsible for unlocking/locking the door when a request is sent but the *ServerTCP* thread.

The shared access to the GPIO pins is handled using critical sections and a *mutex* flag provided by the C++ mutex class.

The main threads of the application at the server side are described in Fig. 3 and 4. The *Print* thread was primary intended for local testing during development and may be removed in the final version. A new *ServerTCP* thread is open for each client connection to assure multiple sessions. This architecture may be easily extended to handle new sensors and actuators if security features are of concern or new communication module such as GPRS to alert the system administrator by SMS.

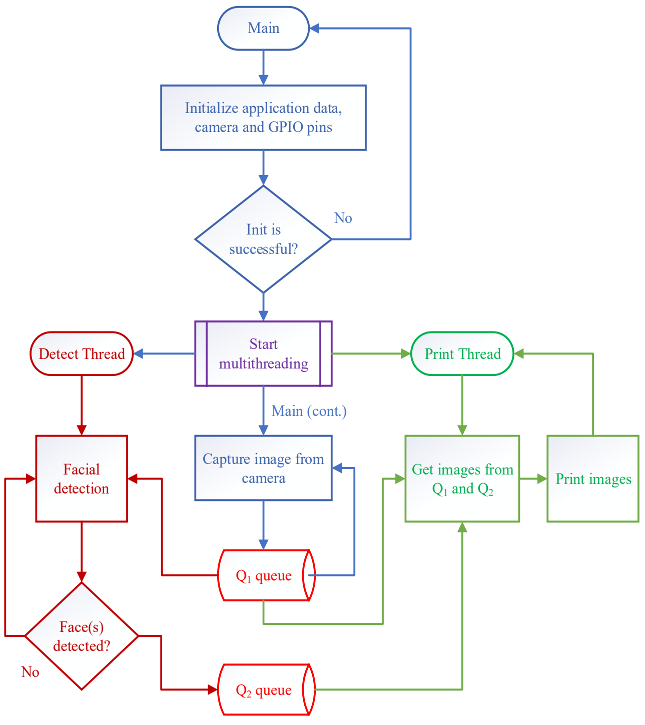
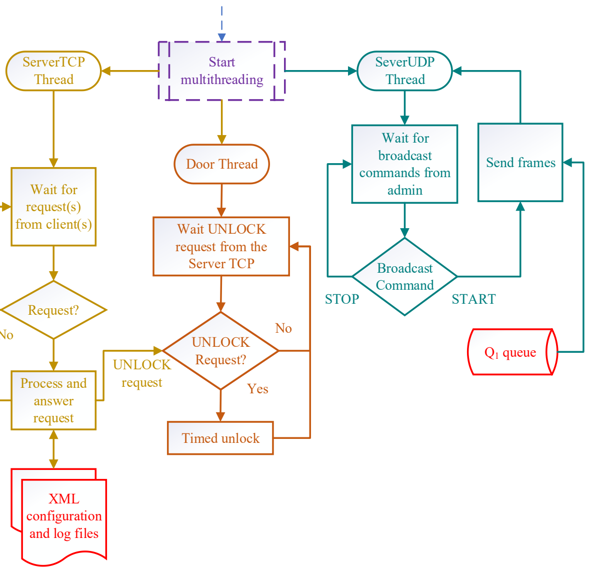


Fig. 3. Implementation of the *Main*, *Detect* and *Print* threads

The embedded controller application features a database of registered users and also a log with the user actions stored in two XML files. This format has been chosen due to the ease of use.

Users are divided into two categories: system administrator and standard user. Standard users have the option of unlocking the door and tracking pictures taken from the camera. In addition, the system administrator may add new users, edit their profile, or delete them (still they cannot delete themselves, so at least the system administrator will remain registered in the application at any time).



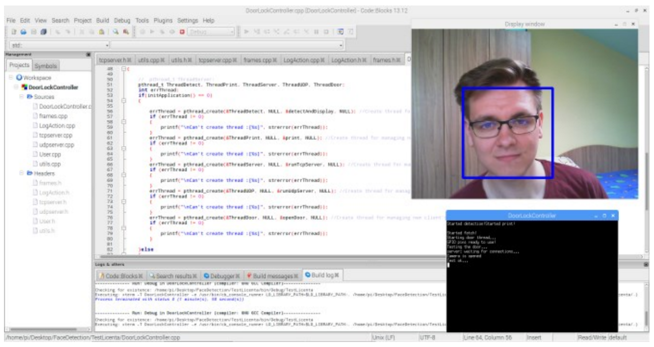
ServerTCP Thread
Start multithreading
SeverUDP Thread
Door Thread
Wait for request(s) from client(s)
Wait UNLOCK request from the Server TCP
Wait for broadcast commands from admin
Send frames
Request?
No
UNLOCK Request?
No
Yes
UNLOCK request
Broadcast Command
STOP
START
Q_1 queue
Process and answer request
Timed unlock
XML configuration and log files

Fig. 4. Implementation of the *ServerTCP*, *ServerUDP* and *Door* threads

B. Use of Video Camera and Face Detection

The OpenCV (Open Source Computer Vision) library was used to retrieve camera images, process them and detect faces. OpenCV is a library of features designed specifically for real-time image processing applications [12]. The library is available on multiple platforms, including Windows, MacOS, Linux, Android, iOS or Blackberry, and is freely available under the BSD open source license.

Launched for the first time in 1999, the OpenCV project was originally an Intel company initiative to advance applications with high processing requirements. The first version was made available to the general public in 2000 at the IEEE Conference on Computer Vision and Pattern Recognition. Currently, the latest release is 3.4.0. The library was developed in C++, but ports for other programming languages such as C#, Java, Python, MATLAB, Perl, Haskell, or Ruby have been developed.



DoorLockController.cpp [DoorLockController] - Code::Blocks 13.12
File Edit View Search Project Build Debug Tools Plugins Settings Help
tcpserver.h utils.cpp utils.h tcpserver.cpp frames.cpp LogAction.cpp LogAction.h frames.h
Logs & others
Code::Blocks Search results Debugger Build messages Build log
Display window
DoorLockController

```
{
    pthread_t ThreadDetect, ThreadPrint, ThreadServer, ThreadUDP, ThreadDoor;
    int errThread;
    if(initApplication() == 0)
    {
        errThread = pthread_create(&ThreadDetect, NULL, &detectAndDisplay, NULL);
        if (errThread != 0)
        {
            printf("\nCan't create thread :[%s]", strerror(errThread));
        }
        errThread = pthread_create(&ThreadPrint, NULL, &print, NULL);
        if (errThread != 0)
        {
            printf("\nCan't create thread :[%s]", strerror(errThread));
        }
        errThread = pthread_create(&ThreadServer, NULL, &runTcpServer, NULL);
        if (errThread != 0)
        {
            printf("\nCan't create thread :[%s]", strerror(errThread));
        }
        errThread = pthread_create(&ThreadUDP, NULL, &runUdpServer, NULL);
        if (errThread != 0)
        {
            printf("\nCan't create thread :[%s]", strerror(errThread));
        }
        errThread = pthread_create(&ThreadDoor, NULL, &openDoor, NULL);
        if (errThread != 0)
        {
            printf("\nCan't create thread :[%s]", strerror(errThread));
        }
    }else
```

Fig. 5. Face detection using Haar identifiers and Viola-Jones algorithm

The main areas of applicability for OpenCV are 2D and 3D processing, face and gesture recognition, robotics, object identifications and localization, augmented reality. OpenCV also includes a machine learning library that contains Bayes Classifier, Artificial Neural Networks in order to support some of the above subjects.

In our application Haar identifiers and the Viola-Jones algorithm were used to detect faces. For reliable detection, the face must be front and vertical. Fig. 5 shows a face detection function test.

Having the face detected is a must condition for the authentication process to step further and request the password. Detected faces are stored when an event log is recorded in the XML file, and can be attached to an email alert or to an SMS if a GPRS is available. As mentioned in section II, a 100% reliable implementation of face recognition in real time on a low cost embedded controller is not available yet and will be considered for the future work.

C. *Electromagnetic Door Lock and Driver Circuit*

The electromagnetic door lock unlocks when the coil of the solenoid is fed with 1.2A at 12Vdc.

GPIO pins of Raspberry Pi3 model B can source maximum 50mA at 3.3V, so a level shifter and power amplifier circuit with two transistors (BC338 and TIP112) and related resistors fed from an external 12Vdc power supply was used to connect the EM coil at the Raspberry Pi3 GPIO pins.

V. THE MOBILE APPLICATION

At the client side, a mobile application was developed in Java using Android Studio 3.0.1 to create the client interface that enables users to control some aspects of the server application. A block diagram of the client side mobile application user interface is shown in Fig. 6.

As mentioned in the previous section, users are divided into two categories: standard users and administrator. This differentiation become very clear in the mobile application because only the system administrator has access to the user area and the XML configuration and log files.

When the mobile application is started, the connection to the server is attempted via the Transmission Control Protocol (TCP) protocol, after which the user is asked for the user and the password (previously added by the administrator). If the connection fails, it will be notified by a dialog. Fig. 7 and 8 show the main and the menu pages for two instances of the application based on the type of logged user.

To open the door, the user must touch the "Unlock" button on the main page (Fig. 7). If their face was detected and stored, for security reasons, it is required to enter the password or use the fingerprint sensor (if the smartphone is equipped with a fingerprint sensor). If authentication succeeds, the ServerTCP thread on the server side sends a request to the Door thread to unlock the door for 10 seconds, as described in the previous section.

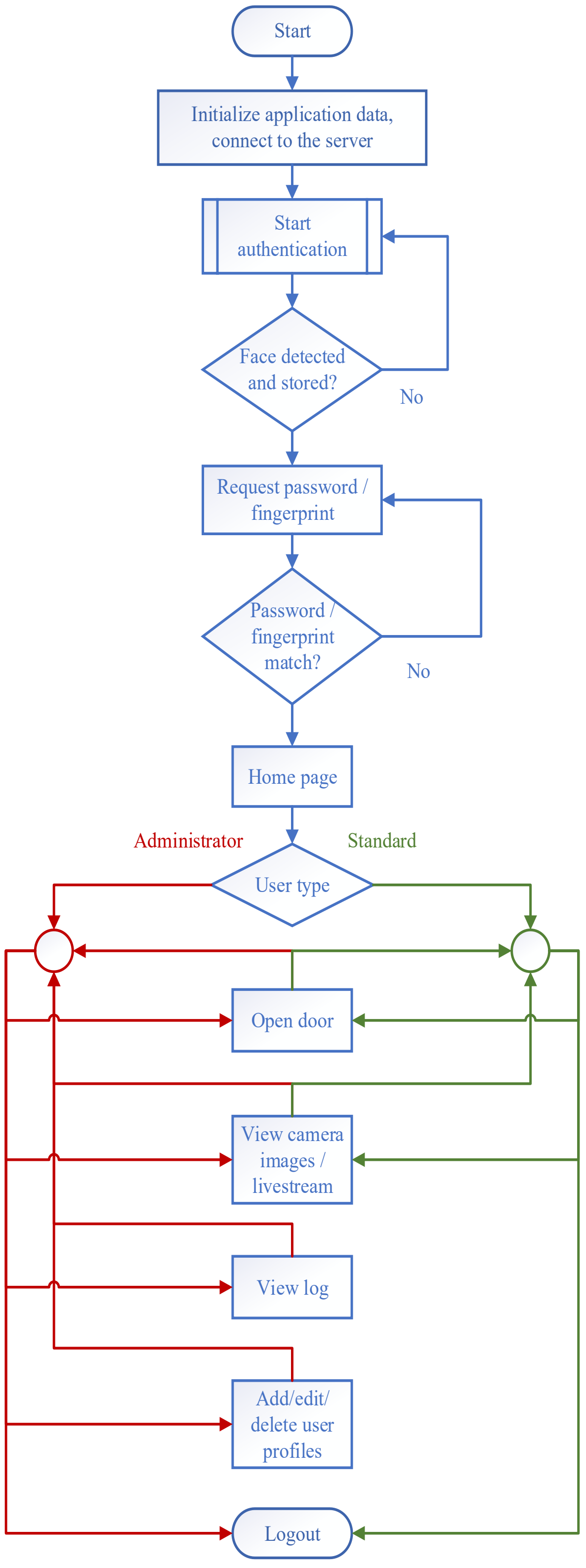

Start
Initialize application data, connect to the server
Start authentication
Face detected and stored?
No
Request password / fingerprint
Password / fingerprint match?
No
Home page
Administrator
Standard
User type
Open door
View camera images / livestream
View log
Add/edit/ delete user profiles
Logout

Fig. 6. Block diagram of the mobile application

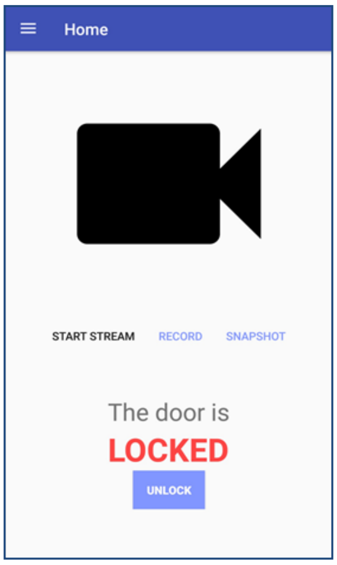
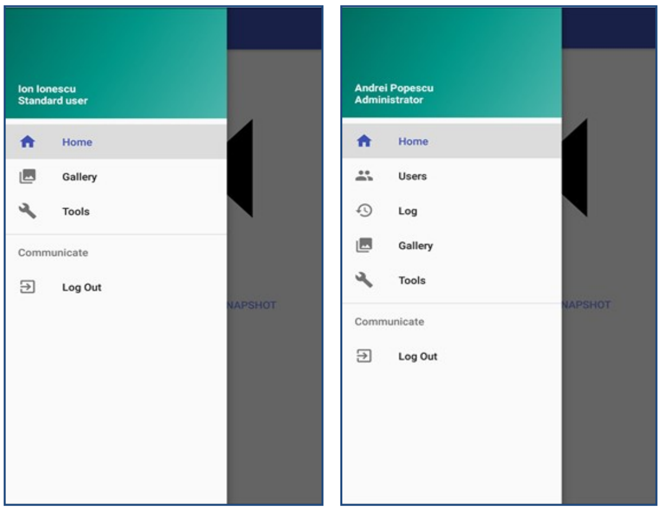


Fig. 7. Main page of the mobile application (client side)



a) *b)*

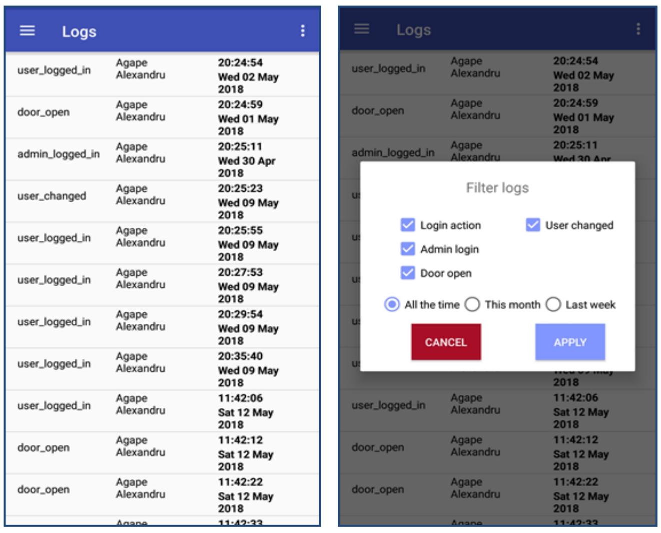
Fig. 8. Sample menu pages: *a)* standard user, *b)* administrator

Administrator can add, edit, or delete users. To access this section, also for security reasons, the administrator will be required to enter their username and password (or use their smartphone fingerprint sensor).

The Log section shown in Fig. 9*a* allows the administrator to track all user actions. There are five types of actions that can be logged and viewed:

- User logged in to the application;
- User unlocked the door;
- User watched images taken from the camera;
- Administrator entered the user configuration area;
- Administrator has modified, added, or deleted a user.

For each action, it stores its type, the name of the user, the date and time at which the action was taken. The administrator may filter the logged data by type and /or date to focus on specific events (Fig. 9b).



user_logged_in	Agape Alexandru	20:24:54 Wed 02 May 2018
door_open	Agape Alexandru	20:24:59 Wed 01 May 2018
admin_logged_in	Agape Alexandru	20:25:11 Wed 30 Apr 2018
user_changed	Agape Alexandru	20:25:23 Wed 09 May 2018
user_logged_in	Agape Alexandru	20:25:55 Wed 09 May 2018
user_logged_in	Agape Alexandru	20:27:53 Wed 09 May 2018
user_logged_in	Agape Alexandru	20:29:54 Wed 09 May 2018
user_logged_in	Agape Alexandru	20:35:40 Wed 09 May 2018
user_logged_in	Agape Alexandru	11:42:06 Sat 12 May 2018
door_open	Agape Alexandru	11:42:12 Sat 12 May 2018
door_open	Agape Alexandru	11:42:22 Sat 12 May 2018

Filter logs

☑ Login action ☑ User changed
☑ Admin login
☑ Door open

[✓ All the time] This month Last week

CANCEL APPLY

a) *b)*

Fig. 9. Sample log pages: *a)* full info, *b)* filtered info

VI. Experimental Results

The custom embedded controller application was developed in C/C++ on a PC running Ubuntu 16.04, Linux kernel 4.13 in Eclipse using the GCC5.4 compiler. This was due to the major difficulties of running debugging sessions on the Raspberry Pi3 target due to the lack of resources (memory and processor power) comparing to those of a PC laptop (i.e. a complete project build takes about 5 minutes on Raspberry Pi3).

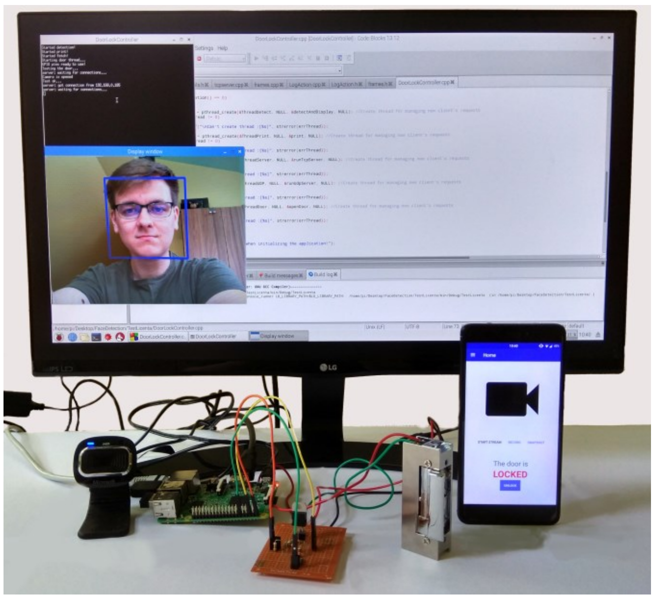


Fig. 10. Experimental setup

For the release version only, the source code was transferred and recompiled on Raspberry Pi3 in Eclipse on Debian 8, Linux kernel 4.14. Socket programming in C was used to implement TCP and UDP servers on ports 3600 and 3603, respectively. The entire experimental setup is shown in Fig.10.

Both parts of the application were tested in a local area network environment and it is no less secure than the local WiFi network and the client smartphone are. The user has to connect to the WiFi network in order to subsequently attempt to contact the host controller and send a door open request. The request is accepted only if at least a face was detected a stored in a host controller file and the correct password or fingerprint was entered. Only trusted standard users that have installed the mobile application, have a valid password to connect to the local WiFi network, and pass the authentication steps of the secure door lock access system may unlock the door.

Any other person attempting to enter the door without the fulfillment of the entire set of aforementioned conditions have their faces detected (if possible) and stored in files on the host controller SD card.

If the host controller, EM door lock and the local area network WiFi router should be fed from an Uninterruptible Power Supply (UPS), the access control system remains active even in case of a temporary interruption of the main power supply, so that registered users will not be affected. Future developments will add new features such as having e-mail and/or SMS alert of the system administrator if there are detected access requests by unknown users and a manual remote door unlock command available only to the system administrator.

VII. Conclusions

An Internet-enabled access control system was implemented using a Raspberry Pi3 model B host controller and a mobile application. Compared to other solutions discussed in the literature, the proposed client-server application for the access control system has the advantages to be smart, flexible enough, and less exposed to unauthorized intruders and breakouts than the classical mechanical or electronic door locks.

Instead of using and embedded web server on the host controller as was the case in [2] and [3], and end devices in an IoT wireless LPAN like in [5-7], custom applications for both the server and client side have been developed communicating through the local WiFi network and using a smartphone as user terminal which is supposed to be secure. There are no threats other than those already existing ones for any computer or smartphone connected to Internet via the local WiFi network. They can be faced using a secured WiFi network, strong passwords and updated software for the WiFi router, the host controller and the user smartphone.

Moreover, the access related events and all the registered user actions are logged and may be monitored on-line or off-line, as in a true security surveillance system, but with an active part that can be further developed and improved.

Raspberry Pi3 model B has plenty of resources to run the host controller application, and the fact that it uses the same class of operating system (Linux) as the PC turned out to be important during development. So, the Eclipse IDE and source code were fully compatible and the debugging sessions could be done on PC when they failed on Raspberry Pi3.

Whether the IP of the host controller is local or public, users can connect to it only within the area of the local WiFi network, or over Internet. Using local connections only hide increases the local host controller, and therefor limits the features of the access control system. On the other hand, making the IP address of the host controller public, increased security should be provided by its local operating system: a firewall and antimalware solutions should be used in order for the whole system to remain safe and trustworthy.

References

[1] S. Sruthy, G. N. Sudhish, “WiFi Enabled Home Security Surveillance System using Raspberry Pi and IoT Module”, Proceedings of IEEE International Conference On Signal Processing, Informatics, Communication And Energy Systems (SPICES), Kollam, 2017

[2] G. Sowjanya, S. Nagaraju, “Design and Implementation of Door Access Control and Security System Based on IoT”, Proceedings of International Conference on Inventive Computation Technologies (ICICT), pp. 83-86, Coimbatore, 2016

[3] A. Hong, C. Xuebin, “Research on embedded access control security system and face recognition system”, in Measurement, vol. 123, pp. 309-322, 2018

[4] Q. Al-Shebani, P. Premaratne, P. Vial, “Embedded Door Access Control Systems Based on Face Recognition: A Survey”, Proceedings of the 7th International Conference on Signal Processing and Communication Systems (ICSPCS), Gold Coast, 2013

[5] M. Sahani, C. Nanda, A. K. Sahu, B. Pattnaik, “Web-Based Online Embedded Door Access Control and Home Security System Based on Face Recognition”, Proceeding of International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, 2015

[6] W. Shao, F. D. Salim, T. Nguyen, M. Youssef, “Who Opened the Room? Device-free Person Identification Using Bluetooth Signals in Door Access” Proceeding of IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, pp. 68-75, 2017

[7] I. A. Khromov, A. A. Dvornikov, “Wireless Access Monitoring and Control System Based on Intrabody Communication”, Proceedings of IEEE Conference on Quality Management, Transport and Information Security, Information Technologies (IT&MQ&IS), Nalchik, 2016

[8] S. Shavi, “Secured Room Access Module”, Proceedings of International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, pp.1134-1138, 2017

[9] A. I. Lita, D. A. Visan, A. G. Mazare, L. M. Ionescu, “Door Automation System for Smart Home Implementation”, Proceedings of 23rd IEEE International Symposium for Design and Technology in Electronic Packaging (SIITME), Constanta, pp. 357-360, 2017

[10] N. A. Hussein, I. A. Mansoori, “Smart Door System for Home Security Using Raspberry pi3”, International Conference on Computer and Applications (ICCA), pp. 395-399, 2017

[11] D, Molloy, Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux, Wiley, Indianapolis, 2016

[12] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, J. Saragihr, R. Shilkrot, Mastering OpenCV 3, 2nd Edition, Birmingham, 2017