Argus: Hardware and Software System for Automatic or Semi-automatic Photo Taking

Zsolt Szécsi
Babeş-Bolyai University
Codespring LLC
Cluj-Napoca
Email: szecsi.zsolt@codespring.ro

Károly Simon
Babeş-Bolyai University
Codespring LLC
Cluj-Napoca
Email: simon.karoly@codespring.ro

Levente Szélyes
Codespring LLC
Cluj-Napoca
Email: szelyes.levente@codespring.ro

***Abstract*—The article presents the Argus project. The aim of the project is the development of an automatic photography system.**

Argus is built on a special hardware system (central server, mini computers with NFC card readers and cameras) and it is controlled by a specific software system. The software system contains a server, a web application and a desktop application, which is running on a Raspberry Pi 2 mini computer.

The architecture of the system is general, so it can be used by companies from different domains. Photo taking is triggered by NFC cards, provided as tickets for different locations (adventure parks, ski resorts, etc.).

As opposed to other similar systems, Argus is built from well-known, inexpensive hardware components, available to everyone on the market.

I. Introduction

The name of the project, Argus, comes from the greek mythology [1]. Argus was a giant with one hundred eyes, who saved many times his home, Arcadia. When Zeus had a relationship with Io, his wife, Hera, abducted Io, making Argus her guardian. Based on the order of Zeus, Hermes killed Argus. In tribute of Argus, Hera took his eyes for decorating the tail of the peacock.

The Argus system tends to be a new giant with many eyes: an automatized system of cameras, recording some of the best moments of its users.

In several times people are in situations when they want to take a photo, but they do not have the possibility to do this. The reason can be a regulation (e.g. in the case of an unsafe location) or a physical difficulty (e.g. caused by a protective equipment), etc. In similar situations a solution can be provided by automatic photography systems, like Argus.

An example for this situation can be the following: some friends go to an adventure park. They want to take pictures while they are climbing, but this is not possible, because the regulation of the adventure park forbids the cameras and mobile devices on the ropes. The reason is the safety of the climbers. Another example can be a ski slope, where the usage of the camera is also dangerous.

There are other camera systems, for similar situations. For example, Skiline[1] is an automatic photography system used at ski resorts. Another similar system is Hexo+[2], which is working with drones. As opposed to these systems, Argus is built on well-known, inexpensive hardware components and it can be easily purchased and installed by companies.

If a company decides to use the Argus system, NFC (Near Field Communication) card readers and connected Wi-Fi cameras will be installed to the checkpoints within a specific location (e.g. adventure park, ski slope, etc.). When a visitor wants to take a photo, he will touch the NFC card reader with his ticket, and the Argus system will take the picture. Photo taking can be triggered based on a predefined scheduling (configurable on a web interface) or by a specific sensor. The photo will be uploaded to the Argus server. The user can check his photos on the Argus website. Here he has the possibility to access his photos without registration, by indicating the identifier of his NFC card (printed on his ticket). If he registers himself into the system, he can see his photos from different locations ordered into albums. The photos are associated to users using the NFC card identifiers. When an NFC card is linked to a user, the photos related to its identifier will be available only for this user.

At the checkpoints, the NFC card reader is attached to a Raspberry Pi mini computer, which is communicating with the cameras connected to the checkpoint and with the central server. A desktop application is running on these Raspberry Pi computers, for configuring location- and checkpoint-specific system parameters (e.g. local Wi-Fi settings, etc.).

The Argus project also publishes a web interface for company managers. Here the checkpoints and cameras can be listed and configured (e.g. the scheduling of the cameras). The system also contains a web interface for system administrators, for system monitorization and user (company) management.

In the first section of the article the hardware elements of the Argus project are presented. The second section presents the tools and technologies used during the development process. The third part of the article describes the implementation of the project. Finally, some conclusions and further development possibilities are presented.

[1]Skiline's homepage: http://www.skiline.cc/

[2]Hexo+'s homepage: https://hexoplus.com/

II. Hardware

The Argus system works with some special hardware elements to realize the photo taking. These hardware components are listed in this section.

A. GoPro Hero 3 White

The "eyes" of the system are GoPro Hero 3 White (Figure 1) cameras. It is a camera designed to be used during extreme sports. It has a waterproof case, which is a big advantage in the case of the Argus system, too. This model is equipped with a wireless adapter and it publishes a wireless hotspot.

Fig. 1. GoPro Hero 3 White camera

The camera runs a web service, which can receive commands. These commands are the same for all GoPro Hero models. In this way, if in the future a manager would like to use a better camera with the Argus system, he can change it with a new model, without any problems.

B. NXP Explore NFC

The NXP company is one of the pioneers of the NFC technology. The NXP Explore NFC board (Figure 2) is developed for Raspberry Pi computers, it can read and write NFC cards.

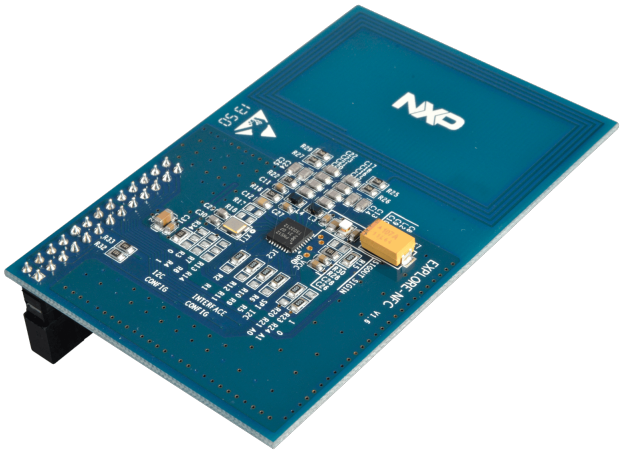

Fig. 2. NXP Explore NFC board

After the installation, the developer can handle this board using C or Python programming languages.

C. Raspberry Pi 2

The camera and the NFC card reader board is controlled by a Raspberry Pi 2 mini computer (Figure II-C). This device can handle multiple types of sensors, so it could be also a good solution for further development possibilities.

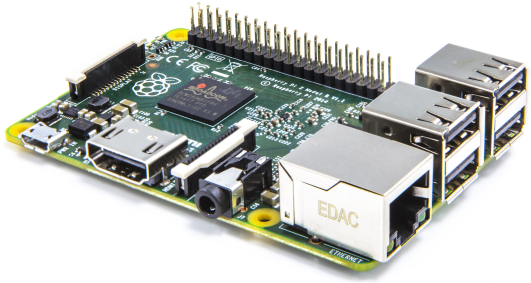

Fig. 3. Raspberry Pi 2 mini computer

This computer is cheap and powerful (900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, 4 USB ports, 40 GPIO pins, Full HDMI port).

It can be used as a PC, so the configuration of the devices is easy with a desktop application running on these mini computers.

D. Edimax Wireless adapter

The connection between the components is resolved through wireless networks. The Raspberry Pi computer communicates with the GoPro camera and also with the Argus server. In this way, this device is connected to two different networks. One is the wireless hotspot created by the GoPro camera, and the other is the wireless hotspot used at the given location, providing internet access.

Fig. 4. Edimax wireless adapter

The Edmix wireless adapter (Figure 4) is designed to be compatible with the Raspberry Pi. The Raspberry Pi computers used within the Argus project are equipped with two or more wireless adapters (one adapter for connecting to the local Wi-Fi network and one adapter for each camera controlled by a given Raspberry Pi).

III. Tools

The Argus project uses the Mercurial distributed, open-source version control system.

The software components of the project are implemented in different programing languages. The Gradle build and dependency management system is used in the case of the Java modules, which is based on the Groovy script language.

The "building" of the JavaScript files, which means repeated actions, like minifying, generating CSS files and copying of the configuration files, is resolved with Grunt, which is a JavaScript task runner. The dependency management for this part is resolved with npm and Bower.

The used project management system is Jira.

On server side data is managed using a MySQL database. On the desktop client side, the caching mechanism is resolved using text files. Here the data is saved in encrypted format.

The generation of the basic server project was resolved with JHipster [2]. Together with a Yeoman generator, this tool can generate a skeleton implementation for a server system, based on the Spring framework (Spring IoC, Spring DATA, Spring Web and Spring Boot) and a web application based on AngularJS and Bootstrap.

The development environment was the Android Studio. It has a very strong Gradle support and it could be helpful in the future, if the project will be extended with a mobile client application.

IV. Technologies

The Argus project is developed in Java, JavaScript, HTML and Python programming languages.

The server is based on the Spring framework [3], [4], [5]. This framework resolves the connection with the database via Spring Data JPA. The dependencies between the components are resolved with Spring IoC. The REST interface is implemented using Spring Web and the REST client is also implemented using the Spring framework.

The Argus project uses hash identifiers in the URLs, like the YouTube system. This feature is implemented using the HashId library.

The web application is developed using the AngularJS and Bootstrap frameworks. AngularJS is a JavaScript MVC framework for web applications. With Bootstrap responsive web user interfaces can be easily implemented.

The Raspberry Pi client is also developed in Java. The communication with the GoPro camera is resolved using Apache Http client. The handling of the NFC card reader is resolved with a Python script, which uses the nxppy library.

Time-related information is handled with Joda time.

The project includes the Swager UI library. It generates a useful documentation for the REST interface, also giving a possibility for testing different requests from a web interface.

V. The Argus project

A. Requirements

The Argus project can be separated into three modules: the Argus server, the Argus web application and the Argus desktop application.

The Argus server works like a remote database. It stores data and it handles user authentication and authorization. The requirements for this component are the following:

- Authenticate the users.
- Ensure a stateless interface, which can be used by different clients.
- Limit the data access properly for different user roles.
- Order the photos into albums based on the NFC card identifiers.
- Link the NFC cards to users and do not grant access to the related data for other users.

The Argus web application gives the possibility for the users to check and to download their photos. This web application is used by the manager to create companies, places and checkpoints, and to configure cameras for these checkpoints. The system administrator also uses this interface to check the state of the server. The requirements for this component are the following:

- Add a possibility for the user to register into the system.
- Add a login possibility for the user.
- Display photos linked to one NFC card, without registration.
- Display albums for the registered user.
- Add a possibility for a registered user to create a new album using an NFC card identifier.
- List the companies controlled by a manager.
- List the places owned by a company
- List the checkpoints and cameras within a given place.
- Provide checkpoint/camera configuration possibilities for the managers.
- Display the state of the server for the system administrator.
- Provide user management support for the system administrator.

The Argus desktop application is running on the Raspberry Pi mini computers. It helps the company managers to configure the system and after the configuration it handles the NFC card reader, it triggers the photo taking and it uploads the photos to the Argus server. The requirements for this component are the following:

- Add a possibility for the manager to configure the credentials for the uploader user.
- Add a possibility for the manager to setup a camera.
- Test the connection with the camera.
- When the Raspberry Pi is connected to the camera, create a camera entity on the server.
- Add a possibility for the manager to reconfigure the local system.
- When a user touches the NFC card reader with an NFC card, trigger a photo taking (based on a pre-configured scheduling).
- Upload the new photo to the server, together with the NFC card identifier.

B. User roles

The Argus web application gives possibility for unregistered users to check and download photos linked with an NFC card identifier.

There are four roles for registered users: uploader, user, manager, administrator.

The uploader user is created for the Raspberry Pi computers. Only authorized uploader users are allowed to upload photos. This type of user can create a camera and can upload new photos, he does not have access to the other parts of the system.

The simple user is the one, who would like to take a photo with his NFC card, and after that, he wants to check his photos using the web application. He can see his photos organized in albums, and he has the possibility to create a new album using an NFC card identifier.

The manager user represents a company, which uses the Argus system. A manager can be linked to more than one companies. He has the possibility to check his companies, to create new places or new checkpoints under a place. He can check and configure the installed cameras on the web interface.

The system administrator has access to everything in the system, he can check the companies, the places and the checkpoints. He has access to the page, which displays the state of the server and he can configure the settings of the server.

C. Architecture

The Argus server has two main component, corresponding to the architecture diagram (Figure V-C). The Java server application manages the data stored in the MySQL database. It also publishes a REST interface. This interface is used for communication by the web application and the desktop application. The web application serves as a user friendly web interface for the users, for the managers and for the administrators.

The Raspberry Pi desktop application includes two modules: the GoPro API and the Nxp NFC core.

The GoPro API is a Java module, which implements the commands for the GoPro Hero 3 White camera. For example the result of the TAKE_PHOTO command is returned by a callback, so this module does not block the caller application.

The Nxp NFC core is a Java module, which wraps a Python script for handling the NFC card reader hardware. The Python script sends a simple JSON object, which contains the identifier of the NFC card or the details of the reading error. This module converts the received JSON objects into Java objects and notifies the listeners about the NFC card reading.

Another component, which is displayed on the diagram, is the Argus API. This module is implemented as a part of the Raspberry Pi desktop application. It resolves the communication between the server and the desktop client application.

D. Using of the Argus system

The Argus project contains different hardware elements. When a company decides to use the system, it orders the

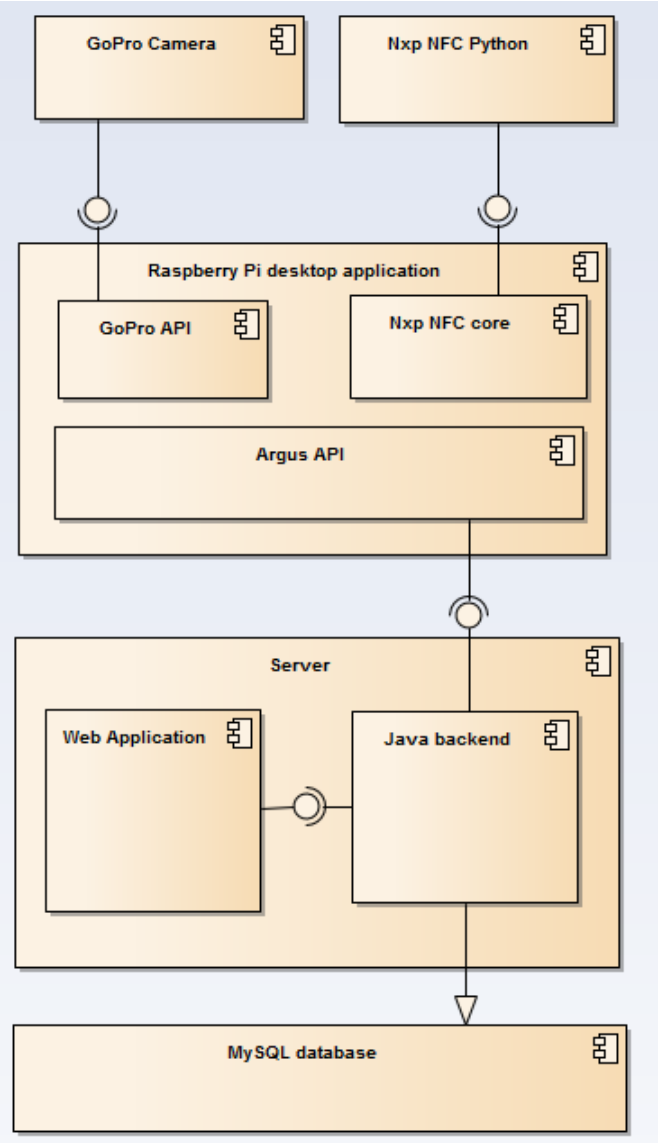

Fig. 5. Architecture of Argus system

Raspberry Pi 2 with wireless adapters and the GoPro camera. The first step is the configuration of the camera. The manager of the company connects the mini computer to a monitor and turn it on. The Argus desktop application automatically starts.

The manager has to log in with the credentials of an uploader user. He logs in into the web application. Selects the company and creates a new place. On the detailed page of the place the manager can create a new uploader user (Figure 6). With credentials of this user he can log into the desktop application.

The desktop application navigates him to an other screen, where he has to configure the camera(Figure 7).

In Argus system every camera is connected to a checkpoint, when the manager starts the configuration, he has to add the identifier of a checkpoint. So goes back to the detailed page of the place (Figure 6) and creates a new checkpoint. On the

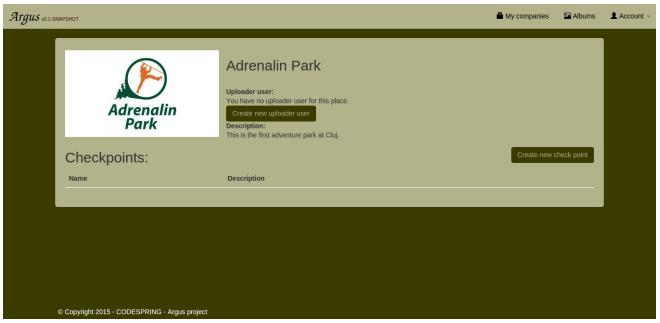

Fig. 6. Argus web application, detailed page of a place

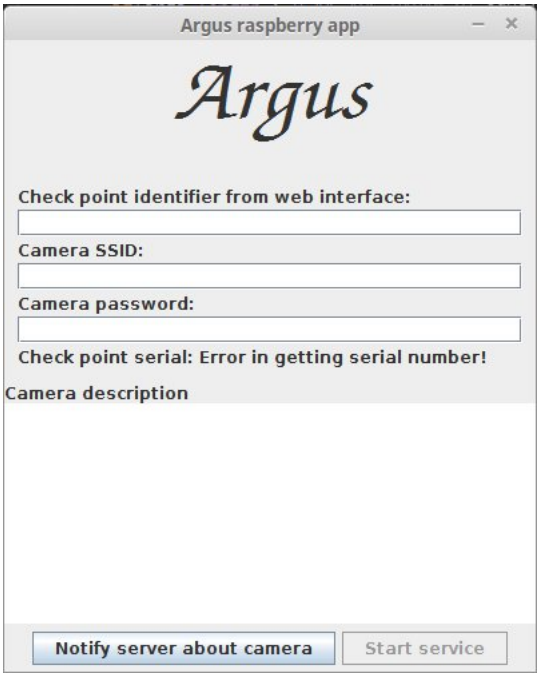

Fig. 7. Argus desktop application, camera configuration screen

detailed page of the checkpoint he can find the identifier of it. He fill the other fields from the desktop application and presses the "Notify server about camera" button. In this moment the desktop application check the connection with the camera and if it is oke, then creates a camera entity on the server.

The local system is configured. The manager can turn off the computer or he can try out the system after then he pressed the "Start service" button. From this moment when the Raspberry Pi mini computer starts automatically the service, which can handle the camera and can upload the new photos. The manager set up the camera on the preferred location and it is ready to use. The delay of the camera id configurable by the manager from the web interface.

The user triggers the system with touching of the Raspberry Pi module with an NFC card. At home he can check and download his images from the web application after adding the identifier of his NFC card (Figure 8).

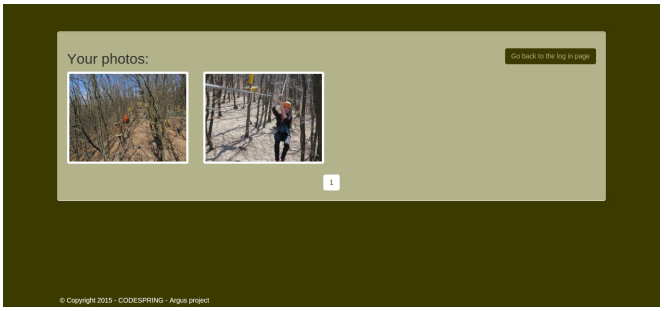

Fig. 8. Argus web application, photos of the user

VI. Conclusions and further development

The current state of the Argus project is a prototype and the development process is still in progress.

The first live test of the system will be in this year in the Adrenalin Park, an adventure park at Cluj-Napoca. During this first test the opinion of the users will be registered for making further improvements.

There are also some known further development possibilities:

- Possibility for managers to configure the system for making videos.
- Delayed photo uploading, which means that the Raspberry Pi computers accumulate the photos locally and upload these only one or two times per day to the server. This mechanism will be useful in locations without a stable internet connection.
- Error reporting: the implementation of a protocol, which will help the camera handler client to repeat a post operation if the connection is lost with the camera. Using this solution the managers can check the state of the system on the web interface.
- Integrating social networks into the system: with this solution the users can login to the system with their Facebook, Twitter or Google+ accounts, and the system will give a possibility to share the photos directly from the web interface.
- Activator application: it could be a tablet application for the ticket seller for specifying the validity of the NFC card. In the current version the validity is always one day. With this new activator application companies can sell promotional tickets, which are valid for more days.

Acknowledgement

This research and development was supported by Codespring LLC.

References

[1] ***, Argus [Online], Available: http://www.britannica.com/EBchecked/topic/34032/Argus
[2] ***, (2015) JHipster Reference Documentation [Online]. Available: https://jhipster.github.io/
[3] Rod Johnson, Juergen Hoeller and co., (2004-2012) Spring Framework Reference Documentation [Online]. Available: http://spring.io/docs
[4] Clarence Ho, Rob Harrop, Pro Spring 3, New York: Springer Science+Business Media, Apress Media LLC, 17 April 2012.

[5] Russ Miles, Tareq Abedrabbo, Michal Bachman, Nicki Watt, Programming Spring, O'Reilly, 2013