Wireless Mobile Networks

2º Semester – 2016/2017

Development of Internet of Things surveillance system (Arduino + Android +SigFox)

Project Report

Group 5

João Francisco Silva	73065
André Stielau	73640

Date: 04/06/2017

Table of Contents

1 Introduction

Our project's goal is to simulate a system with sensors connected to an Arduino. An RFID reader will allow us to distinguish between authorized and unauthorized accesses. A door opening magnetic sensor will be used to send a signal through SigFox, if an intruder is detected, to a cloud based server. This signal will be available to be read by a properly connected Android App.

The initial proposal included a light sensor to aid in the intruders' detection. However, we didn't implement this component.

2 Development

2.1 Arduino

2.1.1 Door Opening Magnetic Sensor

The magnetic sensor was connected to pin D6 of the Akeru on one end and to one of the GND pins on the POWER side of the Arduino.

When the sensor is indicating that the door is closed it outputs LOW and when the door opens it outputs HIGH.

2.1.2 Buzzer

The buzzer was connected to the Akeru on pin D2 on it's positive side and on the negative one to the other GND pin on the POWER side of the Arduino.

It was configured to beep if an intruder was detected.

2.1.3 RFID

The RFID was connected to the Arduino according to the MFRC522 library's recommended setup, as can be seen on table 1.

MFRC522 Reader/PCD Pin	Akeru Pin
RST	D9
SDA(SS)	D10
MOSI	D11
MISO	D12
SCK	D13

Table 1 –Connection between the RC522 and the Akeru

When the Akeru is turned on, it initializes all variables, including the keys to read data blocks on the RFID cards. Our implementation uses the default key values and a card with a different key to be the one to get access denied.

When leaving the "house", the owner of an authorized card will validate his card, activating the sensors.

2.2 SigFox

To communicate with the SigFox backend we used the Akeru library.

On the Uplink side, every time the Akeru sent data to the backend, a pre-configured callback was sent to our webserver, which in turn parses the message, stores the alarm on it's memory and sends a notification to inform the Android Client of the occurred alarm.

On the Downlink side, when the Akeru is powered up and first connects to the network, it sends a request for a Downlink to receive information.

2.3 Web Server

2.3.1 Ngrok - Server exposition service

Since we chose to serve our server in local machines, we used an exposition service so the server may be available to both the SigFox backend and the clients. We did so by running the ngrok application with the arguments 'http' and the port where the server is running in the local machine. This had to be done to both the REST and the WebSockets ports, which default with 8080 and 8000 resulting in two different endpoints that should be manually configured in the mobile application and the SigFox server. The server can also be hosted in the cloud, but we chose not to for development simplicity purposes.

2.3.2 REST - Communication Protocol

The server responds only to REST requests from both the SigFox backend and the clients, that is, they send and retrieve data using only the 'POST' and 'GET' HTTP methods, respectively. There are only three endpoints to this API, '/alarms' to list all the alarms stored and to add a new alarm (via the POST method), '/alarms/:id' to retrieve the alarm with the chosen id, and '/config' to send the authorized RFID IDs to the Akeru on initialization.

2.3.3 Web Socket - Communication Protocol

In order to have real-time bilateral communications between the server and the clients, REST isn't enough and we used an implementation of the Web Sockets protocol called Socket.io. This was used to register the mobile clients and send them notifications to fetch the new alarms right after they are received by the server.

2.3.4 GCM (Google Cloud Messaging) - Communication API

Since the client's notifications library had a tight integration with this service, we implemented the notification service with either Web Sockets or using GCM, although the registration to this service still has to be done through Web Sockets, we found this an interesting technology to use.

2.4 Android App

React-Native application, with a home page, an alarm list page and an alarm detail page for each alarm, with an http connector, a Socket.io client and a GCM listener.

2.4.1 Alarm List

Simple page to display a list of all the alarms. The page requests the alarms by REST each time the page is opened and renders them in a scrollable list, showing only their timestamp. Clicking the reroutes we are taken to the detail page, where more information can be displayed. For demonstration purposes, we gave names to some mock alarms, to see how much more information can be displayed in future implementations.

2.4.2 Notifications

The application has both the GCM listener and WebSocket waiting for messages from the server, when any of them receives one, the device alerts the user with a system notification, which redirects to the new alarm page, when clicked.

3 Project Description

After setting up all the sensors, RFID reader and half wave antenna, we're ready to start testing our system.

The Arduino code was made so that the alarm is set to be looking for intruders once the "owner" checks out of the house by validating the RFID card on his way out. If a sensor is triggered while he is away, the buzzer starts beeping. To stop it, a valid card must be read by the RC522.

Meanwhile, a message was sent using SigFox's network to its backend. There, it triggers a callback function to our REST Webserver which then stores the alarm and sends a notification to the client(s).

4 Conclusion

We implemented all the essential functionalities for the project to simulate an IoT security system, even though we didn't fully test the light sensor and decided to leave it out of the finished project.

The fact we could work directly with these technologies is what made us choose this subject and this project. We enjoyed developing this implementation and, despite the lack of clear and thorough documentation, our work has brought us a better understanding about these subjects. Figuring out how to use the libraries and frameworks by ourselves, most of the time using trial and error approaches, has improved our skills and that is why we consider this project was a success.