Teesside University
School of Computing
2011 - 2012

**Final Year Project**
BSc Computer Science

# Social network library development

Thibaut Havel
L1247434

Supervisor : João F. Ferreira

# Abstract

The main aim of this project is to develop a low-level library that is able to grab and store data from a social network. This library works in an embedded device and stored data had to be used to provide a simple service. To demonstrate the effectiveness of the final library, I created a demo service that interacts with a known social network (for example, Twitter).

# Acknowledgements

I would like to thank my supervisor João Ferreira for his support all over the development of my project.

# Contents

# Development of the library

# 1 Introduction

Our society tends to use more and more social networks (for instance, Twitter or Facebook). At the same time, we are increasingly dependent on the use of embedded devices on a day-to-day basis (for instance, home automation). The goal of this project is to develop a platform that allows an effective and efficient communication between embedded devices and social networks.

The main aim is to develop a low-level library that is able to grab and store data from a social network. This library works in an embedded device and stored data has been used to provide a simple service. To demonstrate the effectiveness of the final library, I created a demo service that interacts with a known social network: Twitter.

One of my personal objectives was to improve my knowledge in low-level development and become familiar with the C language. Also I was looking forward to improving my software development skill while working with a very specific hardware.

Firstly, this report will present the way I started my initial research and how I designed the library according to my functional choices. Secondly, it will present how I've implemented these functionalities, and what I did to test my library. Finally, this report will end by an evaluation of the final product regarding the design choices, the way I've implemented them, and his reliability.

# 2 Methodology

## 2.1 The initial project scheduling

In my project specification, I set my schedule as following:

- January: Research (2 weeks), design (2 weeks).
- Febuary: Design (1 week), Implementation (3 weeks).
- March: Implementation (3 weeks), Testing (1 week).
- April: Evaluation and report compilation (2 weeks).

My supervisor João and I met every week or every two weeks to discuss about planning and progresses of my project.

## 2.2 The effective approaches

My initial approach was to become familiar with the embedded device technology. I had to find an adapted, a small and a simple operating system to work with, thereby I chose **FreeRTOS**[1] supported from my supervisor.

My supervisor had already used this system and he had developed applications before. He provided me one of his own device running with FreeRTOS. So, thanks to João and his knowledges about this operating system, I had a platform in addition to a massive support from him and the online community to develop my library. As I didn't have any knowledge about FreeRTOS, I read several articles which deals with how it works, and how tasks are scheduled in an real-time way.

As a consequence of this chose and because one the goals of this project is to gain knowledge about low-level development, the library has been build using the C language.

[1]FreeRTOS is a light-weight Real-Time Operating System.

Then, I defined every functionalities of the final applications. Basically, the library's features are simple: it should allow a user to receive and send text statues of a social network. For instance a *message on the wall* in **Facebook** or a *tweet* in **Twitter**. Facebook and Twitter are both well known social networks and after some research about them, I choose to build my library suitable for Twitter because of the solid support for **OAuth** which is a secured protocol to access data. Once again, this choice was supported by João.

At this point, I had to defined how to receive and to send tweets, so I've started by looking for any existing solutions. In the next chapter, I will discuss why I chose to build my own library from scratch only using OAuth.

After this key decision, I've learned how OAuth works and how to register an application on Twitter it in order to access to the data.

As I was now aware about what the tools I will use and the way to use them, I designed the library according to the features I wanted to implement.

The next step of the development was to implement the abstract structure of the library. At this stage, I faced lot of issues concerning the use of OAuth, the C language and its requirements. Thanks to my previous years of studies, I learnt the basics of this language but I read a lot of articles, books and tutorials along the implementation.

Once I finished the draft version of the library, I tested every functionalities receiving and sending tweets over my own account and I improved the reliability according to the results of these tests.

## 2.3 The support tools

To help myself into the research and the development of the library, I used some additional appropriate tools:

- A diary: to keep every relevant informations but also as a memory trail of the development chronology.
- Github[2]: to back up the source code and share my progress with my supervisor.
- FreeRTOS POSIX[3] simulator: to develop and to test my library without any embedded device.

[2]Github is an online project hosting.
[3]Standards to maintaining compatibility between UNIX operating systems.

# 3 Research

## 3.1 Operating System: FreeRTOS

### 3.1.1 Overview

(Quick overview of the system: Free, open source, GP Licence, light-weight)

### 3.1.2 Real-Time System

(Kernel mechanism: priorities and scheduling)

### 3.1.3 Libraries

(Existing libraries: non-free libraries for specific hardware, light system library)

### 3.1.4 POSIX simulator

(Compilation of a library and a task)

## 3.2 Twitter authentication: OAuth protocol

### 3.2.1 Overview

(Common authentication mechanism: token, secret key system, include graphic representations)

### 3.2.2 Existing library in C

(Downloaded and tested library: samples hard to understand, idea: create a simple-to-use library layer)

### 3.2.3 Register an application on Twitter

(Way and proprieties of the registered application)

### 3.2.4 Required libraries

(libcurl: overview and it's seem hard to adapt to FreeRTOS, idea: create a very simple HTTP request library)
(OpenSSL: overview and it's seem hard to adapt to FreeRTOS, idea)

# 4 Design

## 4.1 Goal of the library

(Layer which simplify the authentication, using the adapted libraries of OAuth, libcurl and OpenSSL)

## 4.2 Requisites

(libcurl or any other HTTP library and OpenSSL, both used by OAuth)

## 4.3 Representation

(UML-like representation of the way it will work)

# 5 Implementation

## 5.1 The library

### 5.1.1 Requisites

(What did I do to solve the OpenSSL and Libcurl problem)

### 5.1.2 Simplification layer

(What the programmer need to know before coding: private and public key)
(How le library use OAuth to do the authentication)

## 5.2 A demo application

(Graphic representation of the use of my library layer)

## 6 Testing of the library

# 7 Evaluation of the ptoject

## 7.1 Goal

(Is my goal achieved, why/why not?)
(Is my work could be use by someone else, why/why not?)

## 7.2 Schedule

(Did I follow my schedule, why/why not?)

## 7.3 Improvements

(What is it possible to do to improve my library?)

## 8 Conclusion

# Bibliography

[1] F. Surname, “Title,” 2000.

[2] D. E. Knuth, *The TEXbook*. Addison-Wesley, 1990.