Epita

Book Of Specification

Not a Barcode

EpiCode

Authors:

Stanislas Sokolov

Jack Chouchani

Lucas Martin

Youness Suleiman

January 2018

Table of contents

1 The Project

1.1 Project Synopsis

EpiCode will be like a QRCode, it will be a bi-dimensional matrix, machine readable, that contains information about the item to which it is attached. The project aim will be to develop a software capable of both generating *EpiCodes* and reading them.

Generating an EpiCode will be the processing data part. The input data can be a web address, some text, numbers or any other information, and then it is morphed into an image. On the other hand, we have the reading part, which will involve image processing, as well as decrypting of the information contained in the code.

1.2 Project goals

By developing Epicode we plan to complete the following goals :

- Encoding and decrypting Epicode
- Read Epicode
- Read QrCode

Moreover doing such project will benefit us in the following ways :

- Lot of work on algorithm
- Use of image processing

The project will be developed under the following constraints :

- Language of development : **C**
- Work on : **Archlinux**
- Tolerance : **Zero warning or error**
- Target : **Everyone above 2 years old**

1.3 Task distribution

In order to complete the project as quickly and efficiently as possible, we need to have a solid organization. Thus we decided to divide the work, the following table is to show the task distribution.

	Stanislas	Lucas	Jack	Youness
Optical Rec	✓	✓		
Data Encryption / Decryption	✓	✓	✓	✓
Graphical				✓
Website	✓		✓	

2 Specifications

2.1 Optical recognition

The optical recognition is a critical part of the project. In order to read a Qr-Code or EpiCode we will need to retrieve the code from an image. Such a process is alike what we did in the previous project, namely the Optical Character Recognition.

Thus in order to make a readable EpiCode or to retrieve the EpiCode from an image we need to establish some markings/patterns in the code to allow the program to recognize the portion of the image which contain the EpiCode (or the QrCode).

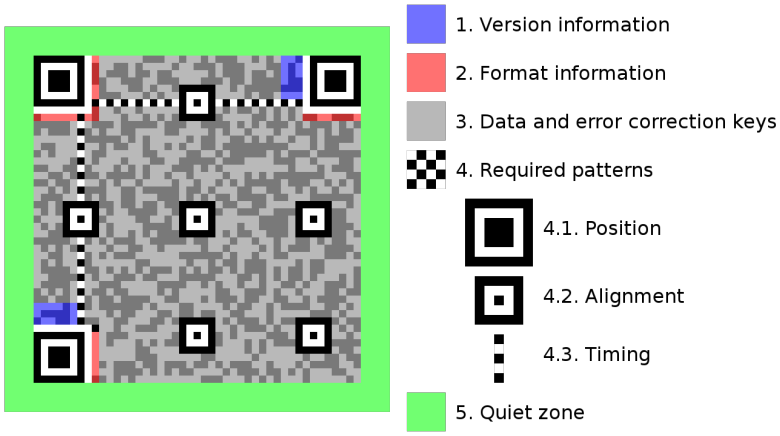

Figure 1: Structure of a normal Qr-Code

We plan to use the basic structure of a Qr-Code to make our own EpiCode's structure. Using this type of marking/pattern will simplify the process of retrieving the code.

2.1.1 Image processing

This part shall be oriented toward the improvement of the image's quality. We will apply a gray level then a black and white level to make the Qr-Code stand out, similar to what we did to extract the text for the OCR.

Moreover the will check if the image is correctly oriented, if the 3 position patterns are correctly positioned on our image (one to the bottom left, one to the top left and one to the top right). If it is not then we will need to rotate the image and maybe skew it. If it's correctly aligned and there is no need of skewing then the image is ready to be segmented then decoded.

2.1.2 Segmentation

The segmentation will be necessary in order to locate the Qr-Code in an image, this section is not really complicated as Qr-Codes already have a format that eases it's recognition by code.

After having a correct image (after the image processing), we will segment the image. The segmenting will result in the program retrieving several important informations. We will first extract the format of the Qr-Code which have a fixed position relative to the position pattern, the same for the version.

After retrieving the version and format we will extract the content which is almost all of what's left. To finish we decode it and return the result.

2.2 Encryption & Decryption

2.2.1 Encryption

In order to create our **Epicodes** we need to have a way to encode data into an image. This process will be called *the encryption*. A powerful algorithm will be needed in order to translate/encrypt our data into a readable code.

To encrypt the data we will need to establish a "language" with which we will translate the data. That language shall be used for the encryption and the decryption process.

To encode the data we will have to perform various step. The first one will be the **Data Analysis** which will determine the most efficient method to handle the data. Then we proceed to encode the data, it's the **Data Encoding** that will segment the data and convert it in the appropriate format.

The third step will be the coding of the **Error Correction** using the Reed-Solomon Error Correction algorithm which will provide error correction code-words to make sure that the program read the data correctly.

The fourth step will be the creation the QR-Code, meaning creating the structure according to the Qr-Code Structure. This process will be called the **Structure Message**.

The fifth step will be the moment when the marks are put down, those mark will direct the reading of the Qr-Code, those patterns/marks are common to all Qr-Codes. We will call this process the **Data Marking**.

The sixth step will be make sure the Qr-Code is readable and to find the best way to make it readable, we use masks, the Qr-Codes have 8 masks, to improve the Qr-Code, this step is known as the **Data Masking**. The last step will be add the **format and version information**. Format version will make the reader able to identify the error correction level and the mask pattern used. The version is used to encode the size of the Qr matrix but are used for rather large Qr-Codes.

2.2.2 Decryption

The decryption process will be easier than the encryption because it will be a reverse operation. By using the same language we will be able to retrieve the data. We will be using SDL to retrieve the Qr-Code and then we'll reverse the encryption process to get the data.

By retrieving the format, the version and the patterns in the Qr-Code we will be able to decode the data from the Qr-Code.

2.3 Interface

The EpiCode UI will be minimalist and user-friendly. We will have a window for reading Epicodes and another for when we are generating them. We plan on using SDL for the Graphical User Interface, and develop an Android application using Android NDK, which let us develop an Android app using the C language.

The GUI allows the user to have access to the features by a 'graphical' way. Thus the user has the opportunity not to use the shell but to use a special window with some different buttons linked with different specific functions.

Actually, in this project, the GUI is not crucial for this project, because it is still usable in the shell. Nevertheless, a such interface permits to make the program easier to run.

It could permit the users to interact with the program, because it shows the different options in a single simple window.

When we thought about how we could organize the different buttons on our window. So we made inquiries about the most typical tips to elaborate such interfaces:

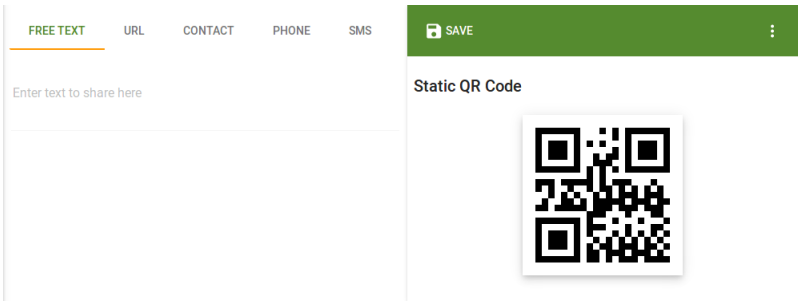

Figure 2: QR code generator template

GTK & Glade In order to realize our GUI (to make the program use more pleasant), we chose to use the library GTK with a software called Glade. GTK is a library which allows us to display different widgets such as text, images, etc... It offers us the ability to have an efficient and a pleasurable interface.

***Bonus:* Android NDK** The Android NDK is a toolset that lets you implement parts of your app in native code, using languages such as C and C++. For certain types of apps, this can help you reuse code libraries written in those languages. The Android NDK will let us develop an Android app using the C language and thus will let us have a mobile application. We might look for other solutions in the future or even choose not to opt for such program after all.

Figure 3: Android NDK

2.4 Website

In order to better present our project, there is a need for a website to showcase all the functionalities. A static web page will be developed, it will contain information about the project, the team and a download page to try **Epicode**.

We will try to keep the website updated with every feature we are adding. The website will be hosted on GitHub and will be freely available for anyone who wants to check about a thing or report an issue.

Figure 4: GitHub Pages

GitHub Pages, as we found that it was the most suitable technology to meet our needs. GitHub Pages is designed to host static web sites directly from a GitHub repository. Alongside GitHub Pages, Jekyll will help to create the base of our page and then the page will be modified to our needs and style.

Figure 5: Jekyll

Jekyll is a simple, blog-aware, static site generator. It takes a template directory containing raw text files in various formats, runs it through a converter (like Markdown) and the Liquid renderer, and spits out a complete, ready-to-publish static website suitable for serving with your favorite web server. Everything is editable which lets us adapt it to **Epicode**.

3 Schedule

In order to work efficiently, the following schedule will keep the team on track on the project completion. It will also provide a good guideline on the advancement of the project in the following semester.

3.1 First Period - From the 19th of January to the 19th of February

The first period will be dedicated at establishing the specific boundaries of the project. It will also see the beginning of the core features of the project.

3.1.1 Specifications

First and foremost, it will be important to define the work synergy in our group, as it is the first time this team has been assembled. Then, the search for information and technologies involved in the project will begin. Afterwards, the projects boundaries will be defined and the work, on a more precise scale, will be divided between the team for efficiency's sake.

3.1.2 Decryption and Encryption

The core elements of the QR code system is of course, the encryption and the decryption, as without them there is no QR code. At first it will be rudimentary, and it will involve a lot of testing and tweaking thereafter.

3.1.3 Website

The Website will be completely finished in order to have a display of the project.

3.1.4 Goals for the first defense

For the first defense, the team would like to achieve the following points:

- Defined guidelines on the project
- Finished and fleshed-out website
- Working encryption and decryption

3.2 Second Period - From the 23rd of February to the 3rd of April

3.2.1 Interface

For a more user-friendly experience, an interface will be given with an easy and clear functioning.

3.2.2 Segmentation and Image processing

The segmentation and image processing is necessary to increase the programs versatility when faced with different situations. In fact, the goal here is to be able to use the Qr code decryption in all cases.

3.2.3 Creating Epicode's bar code model

Here comes the interesting part, where the team will be able to implement it's personal touch to the project. First and foremost, the team will discuss the format of the **EpiCode**, then implement it in order to make it work like QR codes.

3.2.4 Goals for the second defense

For the second defense, the team would like to achieve the following points:

- Working Interface
- Working Segmentation and Image processing
- Finished encryption and decryption
- **EpiCode** implemented, but not fully functioning

3.3 Third Period - From the 6th of April to the 21st of May

The last period will focus on the optimization, and finishing the core features. If time allows it, the team has prepared bonuses that could be added by the end when the project is fully done.

3.3.1 Epicode

By the end of the third defense, the **EpiCode** will be done.

3.3.2 Segmentation and Image processing

Image processing will be the main concern, as the segmentation will be almost done by then.

3.3.3 Interface

The interface will be enhanced in its aesthetic aspect.

3.3.4 Goals for the last defense

Of course for the last defense, the project will be finished. When it comes to the bonuses, they will be implemented only if the team will not be in shortage of time.

3.3.5 Bonuses

Real-Time Decryption RTD consists of using a web-cam or a web-cam like device to give an image to EpiCode in real time and thus making the proccess of decrypting Qr-Codes faster. This option will be added as a bonus if time allows it.

APP implementation The last and most interesting bonus would be the **APP implementation** of the project in C language, as **QR codes** are mostly used on the fly, and what better than a smart phone.

List of Figures

4 References

- https://pages.github.com
- https://jekyllrb.com
- https://www.thonky.com/qr-code-tutorial/introduction
- http://blog.qartis.com/decoding-small-qr-codes-by-hand/
- https://developer.android.com/ndk/index.html
- http://http://libsdl.org