Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Maria Montoya Freire

Visual wireless communications with smartphones

Master's Thesis
Espoo, August 10, 2015

Supervisor: Dr. Mario Di Francesco, Aalto University
Instructor: Dr. Mario Di Francesco, Aalto University

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Maria Montoya Freire		
Title:	Visual wireless communications with smartphones		
Date:	August 10, 2015	**Pages:**	63
Professorship:	Data Communication Software	**Code:**	T-110
Supervisor:	Dr. Mario Di Francesco, Aalto University		
Instructor:	Dr. Mario Di Francesco, Aalto University		

The advent of smartphones has certainly brought many advantages for communications among people. Due to their diverse features, smartphones allow to exchange information in different ways. The most widespread communication technologies for mobile devices – such as WiFi, Bluetooth and Long Term Evolution – exploit wireless transmissions based on electromagnetic radio signals. However, the use of these technologies present some issues with respect to security, coverage and interference.

Recent studies have revealed new approaches for data exchange, such as visible light communication. Using an optical channel for data transmission provides interesting advantages over current technologies: the use of unlicensed spectrum, no interference with radio signal, secure communication and no need for a special infrastructure.

In this thesis, we implemented an application that transfers data between smartphones through visible light communications. To this purpose, the application employs QR codes shown on the display to encode transmitted data and the front-facing camera for receiving the data. We also designed a supporting communication protocol for ensuring a successful transmission between two smartphones. Finally, we evaluated our solution in terms of power consumption and communication performance. The results showed that our solution is suitable for exchanging files of small size.

Keywords:	wireless communication, short-range communication, smartphone, QR codes, barcodes, visible light communication
Language:	English

Acknowledgements

First and foremost, thanks to God for all the blessings received in my life and for giving me the strength to achieve my goals.

To my beloved family: my parents and to my brother and sister for their love and support to follow my dream despite the distance. To all my friends in Ecuador, who accompanied me on this adventure and were always there to help me. To my new friends in Finland, for the greatest moments shared along these two years.

To all my teachers for sharing their knowledge and being always eager to solve my doubts. I would also like to express my gratitude to my supervisor Dr. Mario Di Francesco, for considering me for this thesis topic. All his feedback, ideas and suggestions were helpful to the completion of this work.

And last but not least, I would like to thanks to SENESCYT for giving me the opportunity to have this experience because without their help it would not have been possible to achieve this goal.

Espoo, August 10, 2015

Maria Montoya Freire

Abbreviations and Acronyms

AMOLED	Active-Matrix Organic Light Emitting Diode Display
API	Application Programming Interface
BMP	Bitmap Image File
BLE	Bluetooth Low Energy
CCD	Charge Coupled Device
CP	Cyclic Prefix
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRT	Cathode Ray Tube Display
DOB	Degree of Blur
DPSK	Differential Phase Shift Keying
EAN	International Article Number
ECC	Error Correction Capability
FEC	Forward Error Correction
FPS	Frames Per Second
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HSV	Hue Saturation Value Color Model
IFE	In-flight Entertainment System
IPS	In Plane Switching LCD Display
ISI	Inter-Symbol Interference
LCD	Liquid-Crystal Display
LTE	Long Term Evolution
LED	Light Emitting Diode
NFC	Near Field Communication
OFDM	Orthogonal Frequency-division Multiplexing
OSI	Open System Interconnection Model
OWC	Optical Wireless Communication
PAM	Pulse Amplitude Modulation
PPI	Pixels per inch

QR Code	Quick Response Code
RF	Radio Frequency Technology
RGB	Red Green Blue Color Model
SDK	Software Development Kit
SMS	Short Message Service
TCP	Transport Control Protocol
UPC	Universal Product Code
VLC	Visible Light Communication

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Communication plays a vital role in our lives. People need to communicate between each other for sharing thoughts, ideas or any kind of information. Recent inventions in technology facilitates the communication between people by using tools or devices (e.g., computers, telephones, television, cellphones, network hardware and so on). Usually, these technologies are referred as part of the ICT (Information and Communication Technologies) which include any hardware or software that helps the process of data communication. Especially, the advent of the cell phone was the first step towards the development of a communication era which become even more effective with the emergence of smartphones. While first cell phones included basic features such as voice calls or SMS (Short Message Service), smartphones enable sending emails, making video calls, navigating through the Internet, and so on.

Certainly, the features of smartphones were in possible to achieve without wireless technologies. These technologies use radio waves to enable data communication between devices. This removes the need of wires for data transmission, eventually allowing user mobility [5]. However, their use incurs some issues in aspects such as security, signal coverage. Due to its nature of being wireless, most of these technologies are prone to security attacks which implicates that user data could be exposed. In addition, signal coverage is affected by external factors (e.g., building structure) and interference issues may be experienced when other sources of radio signal are close. Some examples of wireless technologies include WiFi, Bluetooth [6], LTE (Long Term Evolution), NFC (Near Field Communication), etc.

To overcome these issues, new approaches have been considered. One interesting approach includes the usage of optical carriers, for instance Light emitting diodes (LEDs). This type of communication, also referred to as OWC (Optical Wireless Communication) [7], is defined as the use of wireless

communication with optical technology [8]. Based on this approach, new systems have been created that commonly are called VLC (Visible Light Communication) systems.

Moreover, other tools can be used for sharing information. For instance, barcodes, allow the storage of data in a printed symbol. Initially, using barcodes required special optical scanners for obtaining the information encoded in the symbol. More recently smartphones enabled using barcodes without the need for special hardware [9].

1.1 Research topic

Since wireless technologies are prone to interferences (e.g., building materials, electrical devices), communication issues decrease the speed for the transmission to the point that communication could not finish successfully. Some countermeasures can be taken by defining mechanisms to correct possible errors in the transmission. For instance, Bluetooth [10] employs a technique called FHSS (Frequency Hopping Spread Spectrum) to coexist with WiFi in the same environment. But still interference problems exist, which motivates the search of alternatives for short-range data communications.

In recent years, several studies have found the use of optical channels as a possible alternative for data communication [11]. These VLC solutions involve the use of displays, cameras and barcodes. Most of the solutions use custom barcodes based on improving existing solutions by means of a more efficient encoding/decoding process. The obtained results demonstrate the feasibility of using light for data transmission. However, data transmission is mostly limited to one direction; in fact, there are not many studies for scenarios involving a bidirectional communication. The development of this type of communication requires a different approach as it requires a mechanism to control simultaneous transmission and reception at both sides.

1.2 Research goals and methodology

Based on the above-mentioned challenges, we aim to design a VLC solution able to transfer data in two directions by using smartphones. Specifically, we leverage the availability and the capabilities of off-the-shelf smartphones to realize a bidirectional VLC system. To this end, the major goals of this thesis are the following:

- Design a transmission protocol for bi-directional VLC to ensure successful data exchange between devices.

- Develop a proof-of-concept application which uses such a protocol.
- Evaluate the performance of the application in practical scenarios.

The solution presented in this thesis is designed based on the literature review which provided insightful ideas to consider for the implementation and evaluation of the application. The approach used for assessing the application was experimental evaluation [12]. We defined certain metrics (i.e., throughput, power consumption, CPU load and GPU load) to observe and analyse the behaviour of application. The purpose of this work is to provide an alternative solution for data transmission without using a radio network connection.

1.3 Contributions

This thesis presents an Android application whose main feature is to enable a bidirectional communication between two phones for file exchange. To this purpose we used three elements namely, the front-facing camera and the display of the smartphone as well as QR codes. To ensure the correct reception of the file, we required to have a control to coordinate and check the status of the transmission. Since we are using an optical channel for communication, we could not use existing protocols in our application. Therefore, we have designed a protocol, which is inspired by TCP (Transport Control Protocol), according to the channel used.

In addition, our application provides a basic interface to improve user experience for the end users, by reducing the number of steps (i.e., the number of clicks on buttons or selecting options) to perform file exchange, which we do not find in other technologies (e.g., Bluetooth).

Likewise, we evaluated our application to observe the performance of the data transmission in accordance with some metrics. These metrics focused on measuring power consumption, CPU and GPU processing, and throughput. In addition, we performed power consumption measurements by using Bluetooth LE (Low-Energy) to compare it with our application and observe their behaviour throughout the time. This comparison highlights that Bluetooth requires more time for exchanging files as it requires previous steps for pairing devices, whereas our application facilitates the synchronization of the devices and eventually data transmission. The obtained results show that our application ensures files reception and is suitable for exchanging files of small size.

1.4 Structure

The rest of the thesis is organized as follows. Chapter 2 introduces the background about barcodes and several approaches used to enable VLC between devices such as mobile phones and LCD screens. Chapter 3 presents the protocol designed to enable bidirectional communication between two smartphones and the details about the application implemented. Chapter 4 provides the results of application testing by an experimental evaluation of our prototype with focus on communication performance and power consumption. Chapter 5 discusses the results obtained during the experiments and presents the findings of this work. Finally, Chapter 6 concludes this thesis and presents some directions for future work.

Chapter 2

Background

This chapter introduces the relevant technologies used for data communication purposes. We have reviewed the approaches used in several studies to find references and possible ideas for the implementation of our application. These approaches mainly involve the use of different type of barcodes either using existing barcodes or designing new ones in order to enhance data transmission. The principal characteristic of these studies is the fact that communication is possible by using visible light (i.e., visual communication) through the use of cameras and displays.

2.1 Barcodes

A barcode is the representation of data through a printed symbol. Such a symbol is made of geometric shapes (e.g., stripes, squares, dots) with a specified width and size depending on the specific solution. In this context, symbology defines the set of rules (i.e., grammar and character set) to follow when creating a barcode [13]. Using barcodes involves two phases: *encoding* and *decoding*. Encoding refers to the process of converting data into a format suitable to be used as barcode. Conversely, decoding is the reverse process for obtaining the original data from a barcode.

There are two types of barcodes, one-dimensional and two-dimensional, which are described below.

2.1.1 One-dimensional barcodes

A one-dimensional barcode, also simply referred to as *barcode*, is a symbol composed of parallel black lines of different widths, separated by a certain distance, drawn on a white background. Here, one dimensional means that

the height is not considered by the reading process, but just the width of the lines and the distance between them is meaningful.

Barcodes are (and have been) mostly used in supermarkets, e.g., by checkout systems for identifying product packages. In order to read the barcode, it is necessary to use a barcode reader which sends a beam of light over the barcode and determines the amount of light reflected by each color.

As barcodes are used in the industry, there was the need for defining standards, hence, some symbologies emerged. In the grocery industry, there are two simbologies to support only numeric characters: UPC (Universal Product Code) and EAN (International Article Number). The first was created for marking products in the USA and the latter adopted some characteristics of UPC symbology but, with the aim to spread it for international use. Other symbologies include: Code 39, Interleaved 2 of 5 and Code 128, the latter being the standard when encoding alphanumeric characters.

Despite being widely used, barcodes have limited data capacity. Most of the barcodes encode a maximum of 20 characters. Those characters include numeric, alphanumeric and some special characters. Its structure comprises a *quiet zone* to ease scanning process, *start* and *stop* characters to indicate the start and the end of the barcode, *data characters* that represent the data (i.e., "1" for a single-wide bar and "0" for a white space in the barcode) and a *checksum* to verify data integrity [14].

Figure 2.1 depicts an example of a Code 128 barcode encoding the text *"Test barcode"*.

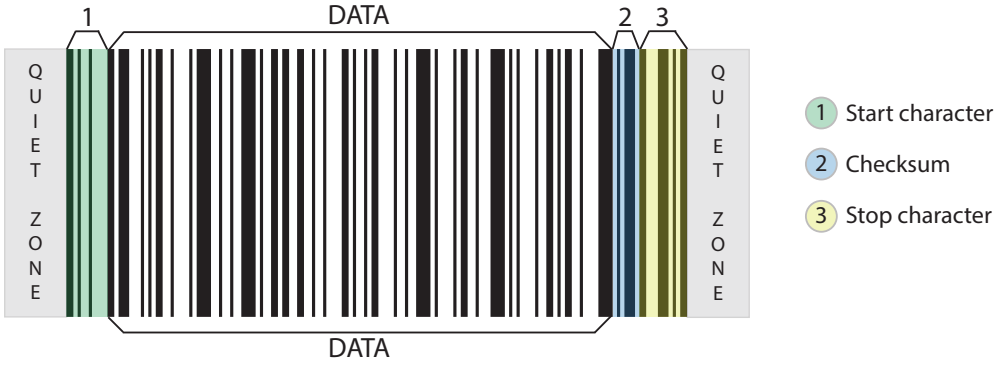
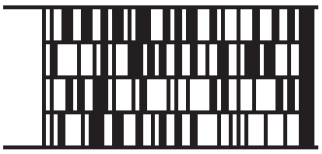


Figure 2.1: Sample barcode

2.1.2 Two-dimensional barcodes

Since the data capacity of one-dimensional barcodes is very limited, the need for a new type of barcode arose. As a result, two-dimensional (2D) barcodes were created to allow the storage of data both horizontally and vertically. There are two main categories for this type of barcode: *stacked barcodes* and *matrix barcodes*.

- **Stacked barcodes** use one-dimensional barcodes stacked on top of each other to store more data in a symbol. The drawback of this approach is the need of more space. Some examples of stacked barcodes include PDF-417, Codabloc k F and Code 49. Figure 2.2 depicts the symbols for these barcodes.



(a) Code 49

(b) CodaBlock F

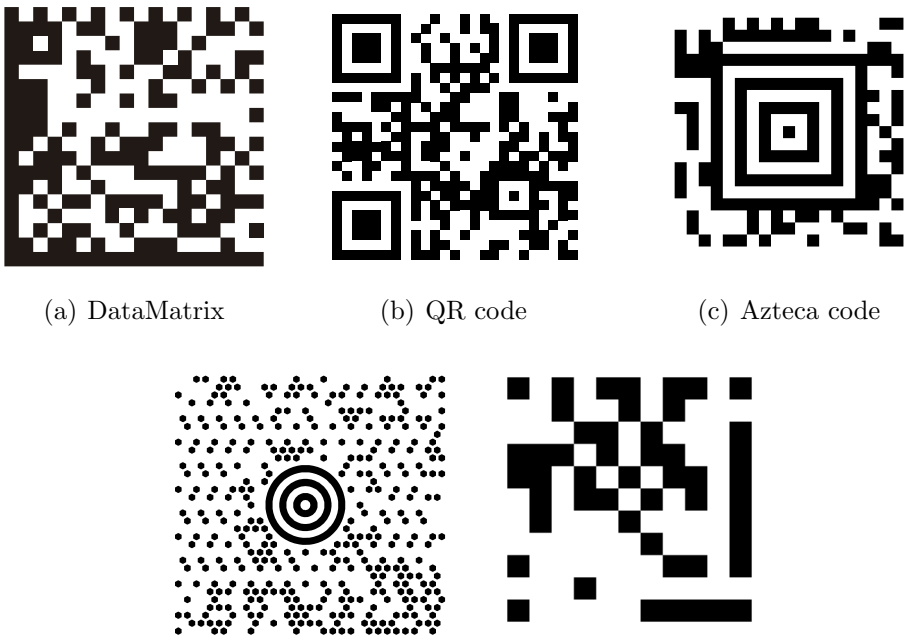
(c) PDF 417

Figure 2.2: Stacked barcodes

- **Matrix barcodes** store encoded data within a matrix by making efficient use of space. The result is a compact barcode with more data capacity. Some examples include DataMatrix, MaxiCode, QR (Quick Response) Code, Azteca codes and EZ codes. In general, 2D barcodes provide more data capacity, better security (since they cannot be easily read by human eyes), and reliability through error correction mechanisms which enabled to spread their usage in different fields [15]. On the other hand, since mobile phones incorporate a camera equipped with a CCD (Charge Coupled Device) sensor, it has become practical to use 2D barcodes. There are two barcodes symbologies mostly used in mobile applications: DataMatrix and QR Code.

The DataMatrix code appeared in 1989 and allowed to store not only numeric values but also text with a data capacity up to 2,000 characters. In 1994 the QR (Quick Response) code was introduced in Japan as a promising solution allowing storage of more data, better mechanism for error correction (i.e., Forward Error Correction or FEC), support of Kanji and Kana characters. They also embedded a position detecting pattern that speeds up barcode scanning. Initially, QR codes were used in the auto industry for tracing car spares but it became very popular in other areas. For instance, they are currently used to store URLs, contact information in business cards, payment information and so on [16, 17].

Figure 2.3 depicts examples of matrix barcodes including DataMatrix and QR codes which are the most common barcodes used in mobile phones.



(a) DataMatrix (b) QR code (c) Azteca code

(d) MaxiCode (e) EZ code

Figure 2.3: Matrix barcodes

Since the focus of this thesis is the implementation of a bidirectional application by using QR codes, then next sections will present more details about such codes.

2.1.3 QR codes

QR codes enable to store different amount of data in the symbol and this is made possible by using versions. A version defines the number of modules (black and white square cells) that can be added to a symbol. There are 40 versions to generate QR codes ranging from 21x21 modules for version 1 up to 177x177 modules for version 40. For each version there are 4 additional modules per side [14, 18].

In addition, the maximum data capacity depends on the data type and the ECC (Error Correction Capability) level. QR Codes support four data type, namely, numeric, alphanumeric, binary and Kanji. The ECC defines four levels to be used in the symbol in order to restore a percentage of the data when the symbol experiences a certain distortion. Table 2.1 shows the maximum data capacity of the symbol (version 40) for every ECC level employed. It is clear that if a symbol uses a higher ECC level, less data can be stored. Therefore, it is important to consider the most important factor when creating a symbol, either more data storage or better data restoration capability [13].

ECC Level	**Maximum data recovery (Approx.)**	**Numeric**	**Alphanumeric**	**Binary**	**Kanji**
L Low	≈ 7 %	7,069	4,296	2,953	1,817
M Medium	≈ 15 %	5,596	3,396	2,331	1,435
Q Quartile	≈ 25 %	3,993	2,420	1,663	1,024
H High	≈ 30 %	3,057	1,852	1,273	784

Table 2.1: Maximum data capacity for QR Code

Structure

The symbol for a QR code mainly involves the use of black and white square cells also called *"modules"* which are placed in a certain position within the symbol for a specific purpose. Figure 2.4 depicts the structure of a QR Code, which is explained next.

- **Quiet zone.** It comprises an area of four wide modules, created with the purpose to ease the recognition of the QR code from its background.

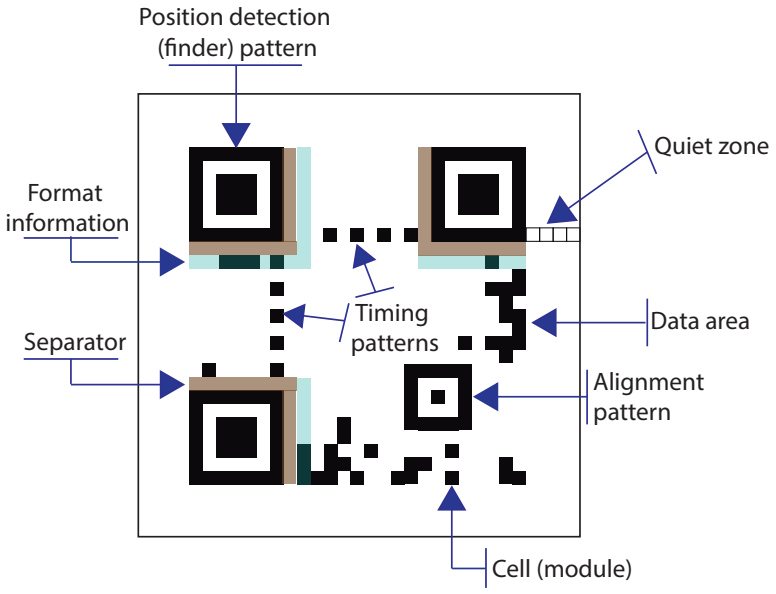


Figure 2.4: QR Code Structure

- **Position detection pattern.** Also called *finder pattern*, involves three equal structures placed at the three corners of the QR code, namely, top-left, top-right and bottom-left [19]. Each structure contains three parts. The first is a 3x3 matrix of black modules enclosed by a second 5x5 matrix of white modules and the last part is a 7x7 matrix of black modules enclosing the previous matrices.

- **Separator.** Separator is a white space area with a width of one cell, placed besides the position detection patterns in order to enhance their recognition.

- **Timing pattern.** It is a sequence of alternating white and black modules that connects the position detection patterns. The timing pattern is used to find the position of each module in the QR code [19, 20].

- **Alignment pattern.** This pattern allows to make any correction in the symbol. It consists of three parts: A 5x5 matrix of black modules, a 3x3 matrix of white modules and a single central black module. The alignment pattern is available in version 2 or greater [19].

- **Data area.** This area refers to the output once data encoding finishes.

At this stage, the error correction codes are included along with the data encoded and a mask pattern is chosen. The purpose of the mask pattern is to improve scanning process by changing color of modules according to certain rules. At the end, the output is a sequence of 0s and 1s placed in the symbol as white and black modules [19, 20].

- **Format information area.** The purpose of this area is to store information related to symbol version, error correction level and the mask pattern used to generate the code [14].

2.2 Visible light communications

Certainly, barcodes have brought many advantages to the industry and end-users by allowing to perform tasks more efficiently. In addition, as technology evolves continuously, the invention of techniques for displays allowed the opportunity to not only print barcodes on paper but to show them also on screens (e.g., television, monitors and mobile phone screens).

Using screens gives opportunity for implementing new applications, for instance, file transfer by displaying data on barcodes. Indeed, this type of communication is referred to as VLC (Visible Light Communication). VLC is defined as an optical wireless communication system which transfers information by making use of the light in a visible spectrum (i.e., 400-700 nm). The advantages of VLC in comparison with the RF (radio frequency) technology are: VLC is not prone to electromagnetic interferences and is a "green" technology since it makes use of existing lighting devices (e.g., Light emitting diodes or LEDs) and at the same time these devices can still be used for other purposes [4, 21, 22].

However, using such screens entails some issues which affect the performance of decoding data. Those issues can be classified in two categories, namely external and internal factors. External factors comprise those issues related to the capture of images while barcodes are displayed on the screen (e.g., blur, ambient light, screen reflection, screen brightness, perspective distortion, distance, rotations) whereas, internal factors involve those issues related to the properties of the barcode (e.g., size, capacity, error correction level, speed for displaying barcodes) [4].

As a result, several studies have been performed in order to resolve those issues with focus on two main objectives: firstly, increase data storage of barcodes; and secondly, adding new features to improve the speed of barcode decoding. Some of these features involve the design of new barcodes based on existing barcodes or modifying barcodes (e.g., DataMatrix and QR code) by

adding more dimensions to increase data capacity or to provide robustness to data transmission.

2.2.1 Modulation-based approaches

The work in [23] presents a different approach to enable data transmission between LCDs and cameras separated by several meters of distance through a system called *PixNet*. Unlike typical solutions that encode data in the visual domain, the novelty of this system is that data is encoded in the frequency domain. To this purpose, *PixNet* uses an algorithm based on the OFDM (Orthogonal Frequency-division Multiplexing) transmission scheme in order to address several issues present during data transmission, namely, perspective distortion, blur, ambient light and frame synchronization.

Since OFDM operates with RF signals, the algorithm requires some changes in order to work properly with LCDs. In other words, output obtained from the algorithm has to be purely real and two-dimensional for displaying on the LCD screens. This can be possible by using a different technique of the one used by OFDM which is a two-dimensional Hermitian Matrix.

Besides, other issues can occur when capturing images on the camera. For instance pixels interfer with pixels of other symbols (i.e., Inter-symbol interference or ISI). To solve ISI, Pixnet makes use of CP (Cyclic Prefix) in order to separate two symbols.

Moreover, the algorithm on the receiver side performs two tasks. The first on corners detection from the captured images in order to obtain the frame. The second task corrects perspective distortion on images by using a sampling principle which consists on finding a relationship for the sampling points between the camera and the LCD and then re-samples such distorted points.

In addition, PixNet includes two more techniques implemented at both sides (i.e., sender and receiver). These techniques are *blur-adaptive coding* and *frame synchronization*. The first technique implements actions to circumvent blur problems by removing higher frequencies, since they cannot be used for data transmission. Those frequencies that present low or moderate attenuation can be used for data transmission but have to be protected with a Reed Solomon error correcting code.

The second technique solves synchronization issues between the sender and the receiver when frames are not properly captured leading to issues in the decoding process, for instance, a captured frame contains two consecutive frames combined or a captured frame made of different parts (e.g., top, bottom) of two consecutive frames. PixNet solves this issue by using

some methods from OpenGL graphics library [24] but also, by controlling the speed at which the receiver is running in order to capture frames correctly from the sender.

The study presents measurement results for PixNet by comparing it with a basic solution that uses QR codes for encoding data. The objective is to determine which one behaves better under certain scenarios. The scenarios involve considering some metrics such as variable distances and view angles. Overall, the results show that the PixNet system can reach higher throughput than just using QR codes. These results are the expected since PixNet has the algorithm to address the issues present when capturing frames. PixNet is able to reach a throughput of up to 12 Mb/s, when distance is 10 meters and it can work with a view angle of up to 120°.

Similarly, Motahari and Adjouadi [25] present another approach with OFDM modulation by using DPSK (Differential phase shift keying) which they called as DPSK-OFDM scheme. By using this scheme, it is possible to improve data transmission by being more tolerant to relative movements of camera. Some simulations were performed in order to compare the performance of their solution with PAM (Pulse Amplitude Modulation) used in 2D barcodes and QPSK-OFDM approach presented in [23]. The results showed that QPSK-OFDM is more efficient since error rates are lower than other approaches.

2.2.2 Multi-dimensional barcodes

In [26], Langlotz and Bimber presents *Unsynchronized 4D barcodes*; a data matrix barcode where data encoding is in 4 dimensions (i.e., width, height, color and time). The main idea of 4D barcodes is displaying a sequence of barcodes on LCD panels and Plasma screens, and recording that sequence with the camera of mobile phones. By adding the time dimension is possible to transmit more information because the data is split into smaller chunks and subsequently converted in an animated image to be shown on the display. As the animated images are an endless loop then, the screen content changes all the time. As a consequence, the decoding process can be affected due to the fact that some data can be missing in the recorded sequence. The reason of this issue is the missing synchronization at the moment of displaying animation and recording. To overcome this issue, the colors are used as a dimension to ensure that most of the data can be obtained from the barcodes. To this end, the RGB (Red Green Blue) color model and their complementary colors (i.e., Cyan Magenta Yellow) are used for the encoding and decoding phases. Those colors determine possible barcode transitions in the sequence. Specifically, inspecting the border colors at the top of the barcode allows to

detect in which color channel the barcode has an inconsistency. The following scenarios could arise: *cyan-red*, *magenta-green* and *yellow-blue*. For instance, a magenta border color at the top means that a code transition in the green channel.

The encoding phase involves data segmentation and the encoding of such segments. Each segment is a frame in the sequence of barcodes. A frame has a header and a 3D barcode. The header contains a field to identify each frame for the purpose of rebuilding the data in the correct order. The 3D barcode is the result of encoding three different 2D barcodes and embedded into each of the RGB channels. Moreover, the decoding phase requires to preprocess the captured frame before decoding. The preprocessing tasks include frame rectification in terms of correcting contrast, brightness, and perspective distortions. After frame optimization, the next step is to identify barcode transitions by inspecting the border color at the top and the bottom of the barcode. If those colors are the same, then it means that the three encoded barcodes are consistent in the RGB color channels; otherwise the inconsistent barcode is discarded. Afterwards, the barcodes have to be converted to grayscale in order to decode them.

Several experiments were carried out to test this approach under certain parameters which are animation speed, barcode size, camera resolution and the option to add data compression. Using this approach between displays (i.e., LCD, Plasma or CRT) and unsynchronized phones is possible to transmit 1400 characters per minute with a success rate of 82%.

The work in [2] presents a VLC system called COBRA (Color Barcode Streaming for Smartphones) which leverages a 2D color barcode to allow real-time streaming between two smartphones. Figure 2.5 depicts the color barcode designed for COBRA system. The color barcode is made of square color blocks of equal size and it mainly consists of three areas: *corner trackers*, *timing reference blocks* and *code area*. The corner trackers allow to rapidly detect the barcode bounds. Once the corners are detected, the timing reference blocks are identified and subsequently used for locating the color blocks in code area. The code area contains the actual encoded data. Basically, the barcode can use 5 colors (i.e., black, white, red, green and blue) for the color blocks but, it is also possible to use other 3 colors (i.e., magenta, cyan and yellow) depending on the image quality.

Since smartphones have small-size screen and low-speed camera, some issues with image quality can occur and decoding process may not be optimal; therefore, COBRA includes some techniques to handle camera limitations and provide higher throughput. These techniques consist of increasing/decreasing the barcode size, blur assessment, and color enhancement. On the sender side, COBRA generates color barcodes (i.e., header, payload,

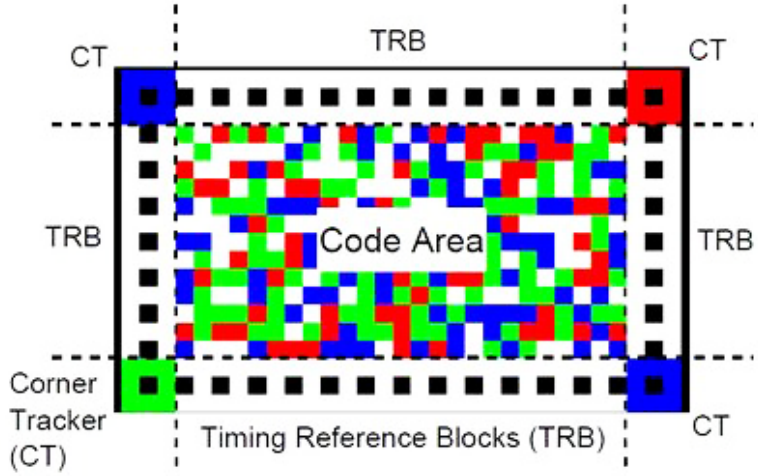


Figure 2.5: Barcode designed for COBRA system [2]

and CRC checksum) from encoded data and then, performs color mapping for each byte from the data. After the mapping process finishes, images of color barcodes are generated and displayed on the screen at a frequency of 15 fps. During this phase, COBRA makes use of accelerometer readings in order to change the barcode size and facilitate image capture at the receiver side. In addition, color ordering is performed to reduce blur.

On the receiver side, two tasks are executed when an image is captured. First, the pre-processing task selects the best image quality based on a metric named as DOB (Degree of blur). The idea is to select such image with less blur for processing. After image selection, there is an color enhancement process for the image based on the HSV (Hue Saturation Value) color model. Second, code extraction task receives the enhanced images and proceeds to detect the corner tracks, timing reference blocks, and the code area in order to decode data.

For measuring the performance of COBRA, the study in [2] considered two metrics (i.e., decoding rate and throughput) under certain scenarios which involve changing factors such as view angle, distance, orientation, ambient light and level of mobility. In addition, COBRA is compared with PixNet [23] for measuring efficiency and overhead for decoding phase. Measurement results demonstrate that COBRA is able to reach a throughput of 91-172 Kbps with a success decode rate of 64% to 98% of the total frames.

2.2.3 Application-specific approaches

Another factor to consider for increasing the success of data decoding is the error correction mechanism employed. In [3], Fath *et al.* present a in-flight application for IFE (In-flight entertainment) systems used in the airplanes cabin by using a different FEC scheme. Basically, their approach consists in encoding the entire data thus obtaining a FEC encoded data stream rather than per frame as commonly used in QR codes. In addition, they aimed at providing an alternative for wireless communication in order to transfer small files such as text or images through the use of a sequence of visual codes. Before obtaining the sequence of codes, the application performs some operations on the file. Firstly, the file is compressed to reduce the number of codes to transmit; since files may contain passenger data therefore, it is also possible to encrypt the file to provide data security. Secondly, the file is encoded with the proposed FEC scheme. Finally, the encoded file is split into several frames to generate the sequence which it shows as an continuous loop on the IFE screen.

Since the data transmission between sender and receiver does not require an initial synchronization, the authors define a protocol to ensure data reception. The design includes three components, namely *payload code*, *meta code* and a *pattern* for frame synchronization. The meta code contains the frame number of the packet in the sequence whereas, the payload code consists of the data for transmission. The solution uses QR codes to encode data while, payload codes employ colors in order to improve data rate. A payload code is the result of combining three single-colored payload codes that represent three successive data packets. The pattern for frame synchronization is used to detect transitions from one frame to another. It makes use of black and white horizontal bars changing frame by frame. There can be two possible scenarios for this pattern: the *even pattern* which is a sequence of black-and-white bars, and the *odd pattern* which is a sequence of white-and-black bars. Figure 2.6 shows the components used in the IFE solution.

The user scenario for the application can be described as follows. It mainly involves two parties a sender which is the IFE screen and a receiver which is the passenger's mobile phone. The receiver starts to record the sequence and once sequence was recorded, the decoder proceed to decode the packets and order them by using the frame number provided in the metadata code. At this point, the result is a FEC encoded data stream therefore, the decoder proceeds to decode again and it obtains the compressed and encrypted data. Finally, file decryption and descompression is performed.

By using the proposed FEC approach along with the colored visual codes, the application can support a data capacity of up to 1158 bytes per frame

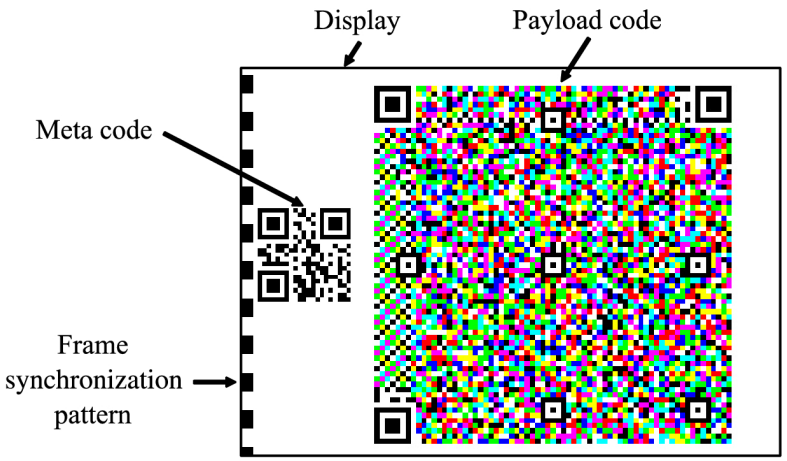


Figure 2.6: Components of IFE application using barcodes [3]

and a bit rate of 120.43 Kbps when displaying 13 frames per second.

Yonezawa *et al.* [27] designed an application called *SENSeTREAM*, which embeds values of several sensors (e.g., temperature, accelerometer, humidity, location, etc.) in QR codes subsequently used as part of a video stream. For instance, an animated avatar can replicate the same movements or gestures of a singer in a live performance by reading the values collected by the sensors. Therefore, users can have a more realistic experience as when they are present in a singer's performance. Basically, the application consists of two components: the *authoring tool* and the *player tool*. The authoring tool, generates a video stream by combining two sources namely, the video captured from several web cameras and the QR codes generated from the collected sensors values. The video is transmitted through live streaming services such as USTREAM and YouTube Live. Once the video is available in the Internet, viewers can start to watch and the player tool will decode the QR codes from the video stream and obtain data by using an API. As part of the research, they performed a pilot study to evaluate the user experience by using SENSeTREAM in an online musical event. The study involves some scale questions to quantify the results. Overall, the application received positive feedback by the users, stating SENSeTREAM as a good alternative to watch online concerts.

2.2.4 Bidirectional communications

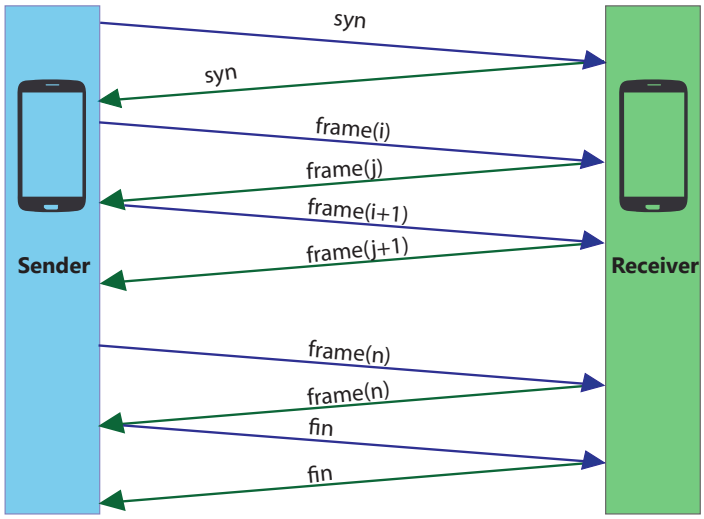
The studies presented so far have only considered communication in one direction, but still is possible to create a communication in two ways. To

this purpose, *CamTalk* [4] is a framework which allows communication between two devices (e.g., mobile phones and tablets) using their displays and front-facing cameras. The framework consists of three modules: the *barcode encoding/decoding service*, the *message transport module* and the *application module*. The service is in charge of encoding data to show on the display and decoding data once images are captured. To this end, QR codes are used for encoding/decoding process. The message transport module allows bidirectional communication between the devices. The application module contains two applications namely, file transfer and key exchange. The key exchange employs Diffie-Hellman algorithm in order to establish a secret communication for exchanging data on an insecure channel.

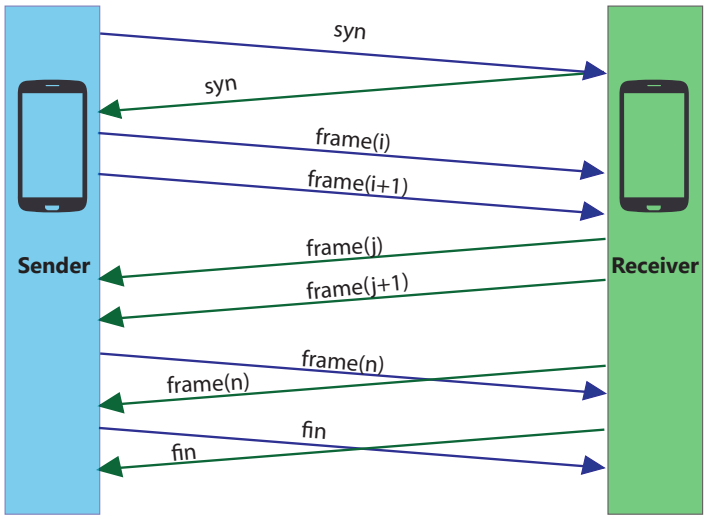
There are two modes for a message transfer: *synchronous* and *asynchronous*. The first mode consists of an initiator that sends a frame (i) to a responder and, it sends the frame (j) to the initiator. This mode requires that both sides send their current frame, otherwise it is not possible to continue with next frame. Figure 2.7(a) depicts the operation of the synchronous mode. The second mode (called batch mode) enables sending multiples frames at the same time from one side to the other. In other words, an initiator can send frames (i) and (i+1) without waiting that responder sends frame (j). Figure 2.7(b) shows the operation for the batch mode.

For data transmission, CamTalk uses frames which are the barcodes shown in the display. Each frame has a header and a payload of a fixed size. The header contains basic information about the frame itself such as the sequence number and the acknowledgement number. The payload refers to the data to be transmitted. There can be two type of frames: control frame and data frame. The control frame is used for the initial connection between the devices. In other words, during the exchange of SYN and FIN frames which are used at the beginning and ending of communication. Also, it allows to notify either to the sender or receiver about missing frames.

The evaluation of CamTalk consisted on testing message transport modes for data transmission between smartphones and tablets. Overall, the batch mode presents better results for transmission in comparison with synchronous mode but under certain conditions. In addition, analysis of factors such as distance, rotations, screen brightness, ambient light, barcode size were performed in the study. CamTalk can transmit at a distance from 11 cm to 22 cm for smartphones and for tablets from 15 cm to 50 cm. It can reach a throughput of 3Kbps by using a Nexus 7 tablet.



(a) Synchronous mode communication



(b) Batch mode communication

Figure 2.7: Message transport modes of CamTalk [4]

Chapter 3

Design and Implementation

This thesis aims to implement an application to enable bi-directional file exchange between two smartphones. To this purpose, a communication protocol was designed in order to set the rules for providing feedback during transmission and also to facilitate file reconstruction when the transmission is over.

This chapter details the system design, the software components used, the protocol design and its operation.

3.1 System design

In the design of the system we need to consider certain aspects to achieve the main goal of this thesis. These aspects refer to the selection of technologies to use and other components which are described below.

- **Barcode selection.** We adopted QR code because of the data capacity provided and the number of libraries available (e.g., ZBar, ZXing) for helping with encoding and decoding processes.
- **Hardware.** The application needs to use two components to enable data transmission namely, the front-facing camera and the display. In this manner, we ensure that both devices capture barcodes, and at the same time show barcodes on their displays.
- **Communication.** The communication between smartphones requires to have a control of the barcodes actually received and missing barcodes. Therefore, we designed a protocol to define the set of rules for the file exchange which ensures a successful data transmission.

Based on these aspects, we have designed and implemented an application that we henceforth refer as *"TransfilesQR"*.

3.1.1 Architecture

TransfilesQR mainly comprises two actors, namely, a *sender* and a *receiver*. Both actors consists of the very same components shown in Figure 3.1. The channel used for communication is an optical channel that can be used through the usage of physical parts of the smartphones (i.e., camera and display).

In order to process the data for generating the barcodes and obtain the data from the barcodes, there are two components namely, *encoder* and *decoder* which leverage third-party libraries for the execution of their tasks.

In addition, there is the frame transfer protocol to keep control of the data transmission on both sides. Finally, the interface for the application to allow the selection of the file to transmit and subsequently start the transmission.

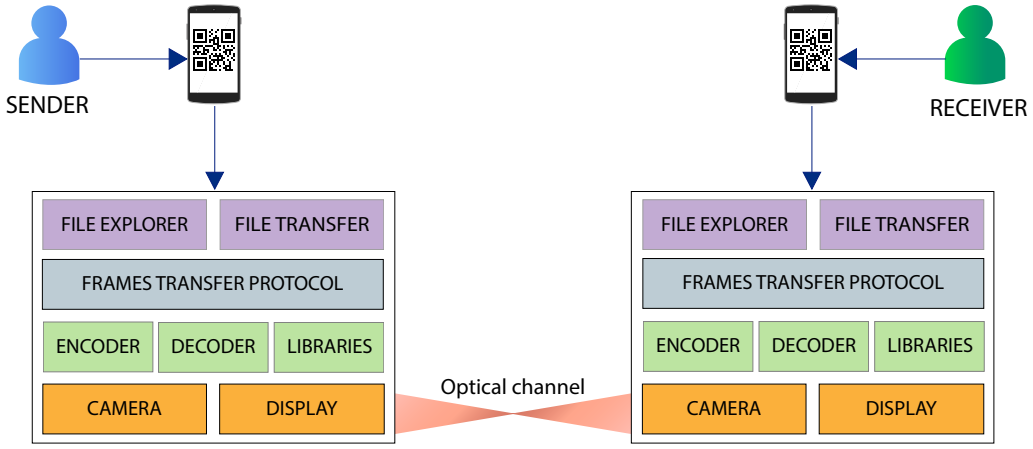


Figure 3.1: TransfilesQR architecture

The *encoder* operation can be seen in Figure 3.2. Basically, the file chosen by the user is the input for the encoder to generate barcodes. Then, the file is split into chunks of a fixed size and metadata (e.g., frame number, total of frames) is concatenated with every chunk that is performed according to the protocol specifications explained in Section 3.2.

For each chunk with metadata, the encoder generates a barcode which becomes in a *frame*. The frames are added to a list that is going to be displayed on the screen of the smartphone at a rate of 10 fps.

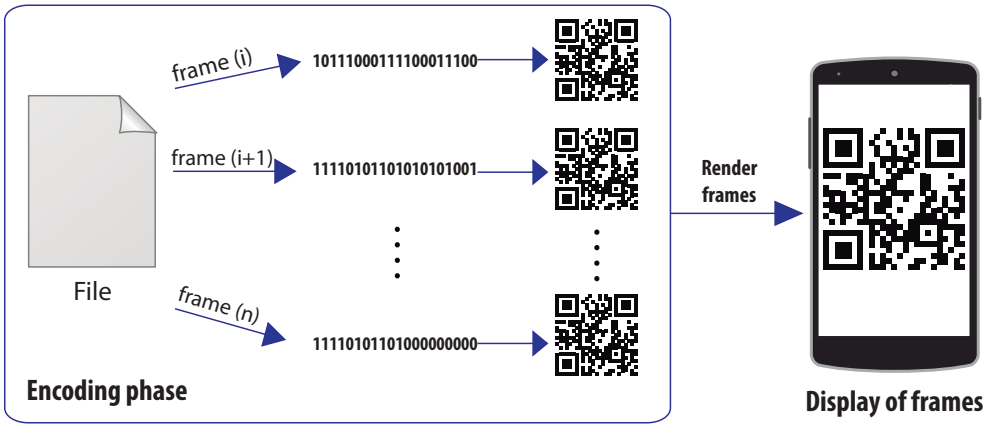


Figure 3.2: Encoding phase

Conversely, the *decoder* is in charge of extracting the data from the frame captured. For each frame captured, the decoder obtains the frame number in order to know the missing frames, and then stores each frame in a list until the transmission is over. Once all the frames are captured, the decoder proceeds to decode frames and place them in order for the file reconstruction. Figure 3.3 shows the decoding phase after all the frames are received.

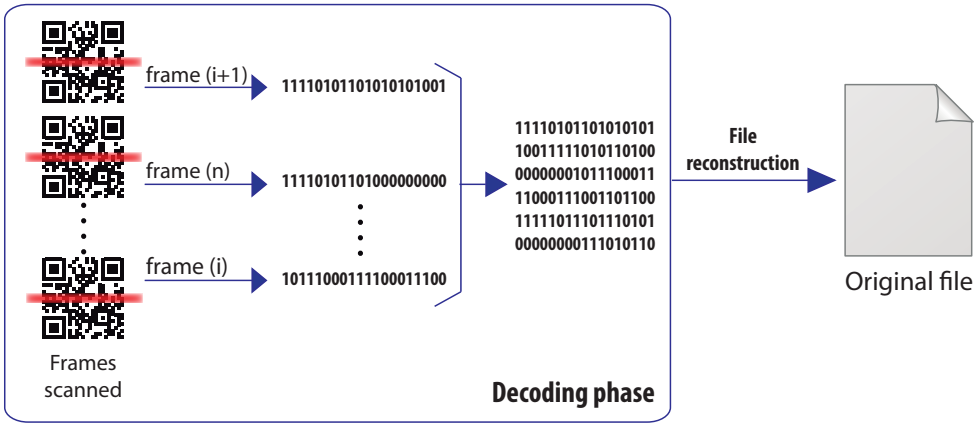


Figure 3.3: Decoding phase

3.2 Protocol design

As TransfilesQR must provide bidirectional communication, it is necessary to design a protocol that controls the frames transmission at both sides of communication by sending notifications for frames received and also optimizing the time for retransmission of loss frames.

There are several protocols that work at different layers of OSI (Open System Interconnection) model for traditional communication channels [28]. In our particular case, we have been inspired by TCP (Transport Control Protocol) [29] for the design of our protocol, which indeed operates at the data link level of the protocol stack.

3.2.1 Frame format

We now present the frame format chosen for every barcode displayed. Basically, the data encoded in a QR code consists of two parts: a header (13 bytes) and a payload (80 bytes). The header includes several fields to control frame transmission, whereas the payload includes a chunk of the file selected by the user. Figure 3.4 depicts the frame format for a QR code, the description of each field is provided below.

- **Frame number (2 bytes).** It contains the current frame number transmitted. This value allows to control number of frames received and also to reconstruct file after transmission is over.
- **Total frames (2 bytes).** Field value used to store number of frames to be transmitted. Since frames may not be received in order, this field is needed to control the number of frames to receive during transmission.
- **Sequence number (2 bytes).** Random number used for establishing a session between the sender and the receiver.
- **ACK number (2 bytes).** The acknowledgment number is used to notify to the other party which frame number was received and also to remove such a frame from the list of frames in order to keep showing on the display those frames that have not been received yet.
- **SYN (1 bit).** Control bit used as a flag (i.e., two possible values 1 or 0) for the session establishment between the sender and the receiver. If the bit SYN is set to 1 then, it means that the sequence numbers have to synchronize.

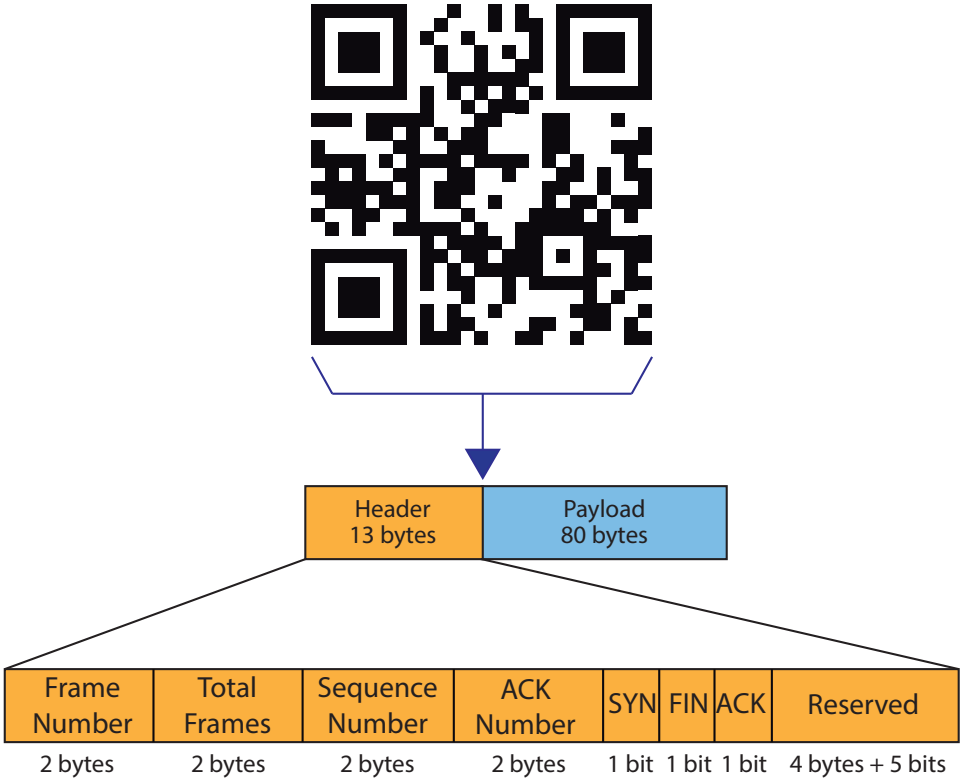


Figure 3.4: Data stored in a QR code

- **FIN (1 bit).** Control bit used as a flag (i.e., two possible values 1 or 0) for a frame used to end a session between the sender and the receiver. If the bit FIN is set to 1 then, it means that the sequence numbers have to synchronize.
- **ACK (1 bit).** Acknowledgement flag (i.e., two possible values 1 or 0) used to indicate that ACK number is set and has to be read. This allows the application to refresh the list of frames by discarding the frame value set in the ACK number.
- **Reserved (4 bytes + 5 bits).** Space reserved for future implementation, for instance adding a window size to make a more efficient use of the communication channel.

3.2.2 Protocol operation

The protocol operation comprises three parts: session establishment, data transmission and session termination. For the session establishment and termination phases there is a *three-handshake* between sender and receiver that involves the exchange of three frames. During the session establishment both sides exchange file name by sending this value as payload. Figure 3.5 presents data transmission between *sender* and *receiver*.

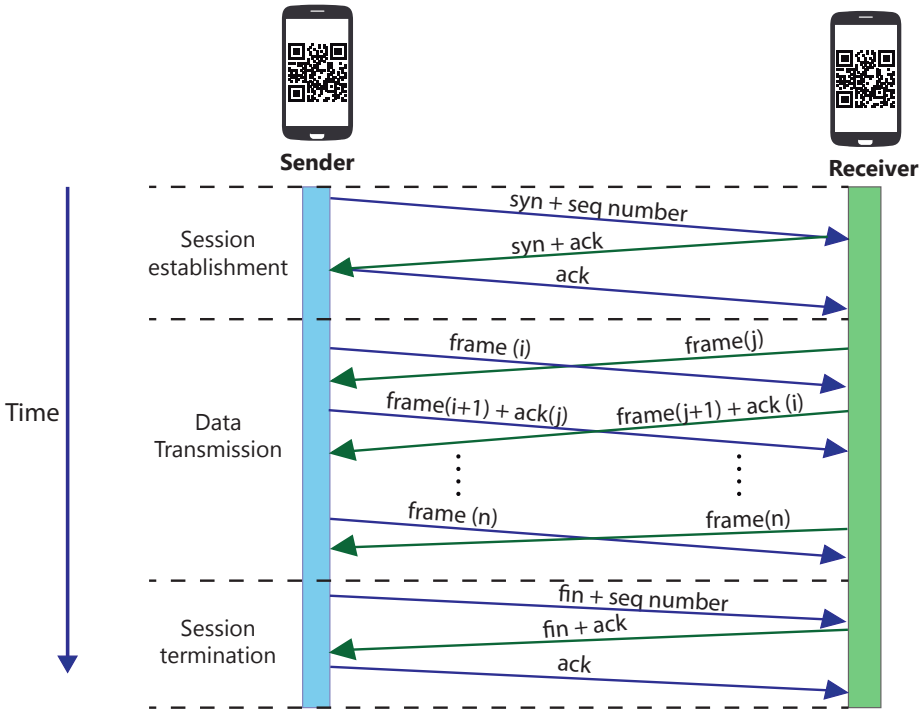


Figure 3.5: Protocol design for TransfilesQR application

In order to transfer files simultaneously, the following steps have to be performed:

1. The *sender* displays a SYN frame, which contains SYN flag set to 1, and a sequence number to be captured by the *Receiver*.

2. The *receiver* scans the SYN frame and obtains the sequence number to create a SYN_ACK frame based on this value. The SYN_ACK frame

contains SYN and ACK flags set to 1, the sequence number from the sender + 1 and the sequence number from the receiver.

3. The *sender* displays an ACK frame to notify the *receiver* that the previous frame was captured and correctly received. Therefore, a session is established.

4. The *sender* and the *receiver* start to display frames from the list and also capture frames from the screens. At this stage, if a frame is captured and correctly decoded, then such a frame will be stored in a list. In addition, the frames belonging to the list displayed on the screen are updated to store in the ACK number field the number of frames received.

   The purpose of this update is to notify the other side that a certain frame has been received, therefore, it has to be removed from the list of frames to be displayed on the screen. This ensures that only unreceived frames are being displayed on the screen.

5. If one of the parties finishes data transmission, it will keep capturing ACK frames until it does not capture more ACK frames from the other party.

6. Once transmission is over at both sides, the *sender* displays a FIN frame which contains FIN flag value set to 1 and a sequence number.

7. The *receiver* scans the FIN frame and obtains the sequence number to create a FIN_ACK frame to be displayed on its screen.

8. The *sender* scans the FIN_ACK frame and creates an ACK frame to notify the *receiver* that all frames have been received. The session is terminated.

3.3 Android implementation

TransfilesQR is built on the android platform and it uses three third-party libraries described below.

- **OpenCV.** Open Source Computer Vision [30] is an open-source library mainly used for image processing in real-time. It offers support to integrate with android applications and supports the Camera API [31].

  In the application, OpenCV is used for the decoding phase. It is in charge of capturing the image from the camera preview and returns a

Mat object. This object is the reference to the image which is later used in the application to obtain data stored in the barcode.

- **ZXing.** Zebra Crossing is an open-source library implemented in Java that is used for generating certain types of barcodes (e.g., 1D or 2D barcodes). It returns a BMP image that is displayed on the screen of smartphone, and also provides functions to obtain original data from a BMP image [32].

  In order to use the library, there are some parameters that need to be configured. Table 3.1 summarizes the values assigned to each property.

Parameter	**Value**
Character Set	ISO-8859-1
Error Correction Level	Q (quartile)
Size	1250 x 1250 (pixels)
Margin (for quiet zone area)	3

Table 3.1: Parameters used for ZXing configuration

- **OpenGL ES.** Open Graphic Library for embedded systems is an API provided by Android that is a subset of the Open Graphics Library (OpenGL). It allows to render 2D and 3D graphics with high performance and is very used for development of games [33].

  TransfilesQR uses this library for rendering the list of frames on the display of the smartphone. Two components are used for this purpose namely, a *GLSurfaceView* and a *GL_TEXTURE_2D* to place a BMP image created on-the-fly. Basically, we first create a square on the display of the same size of the QR code and we place on top of it the texture containing the image.

Overall, we make use of several *Threads*, an *AsyncTask* and a *Handler* along with these libraries in order to perform data processing more efficiently and avoid an excessive computational work on the main thread of the application. More details about implementation can be found in Appendix A.

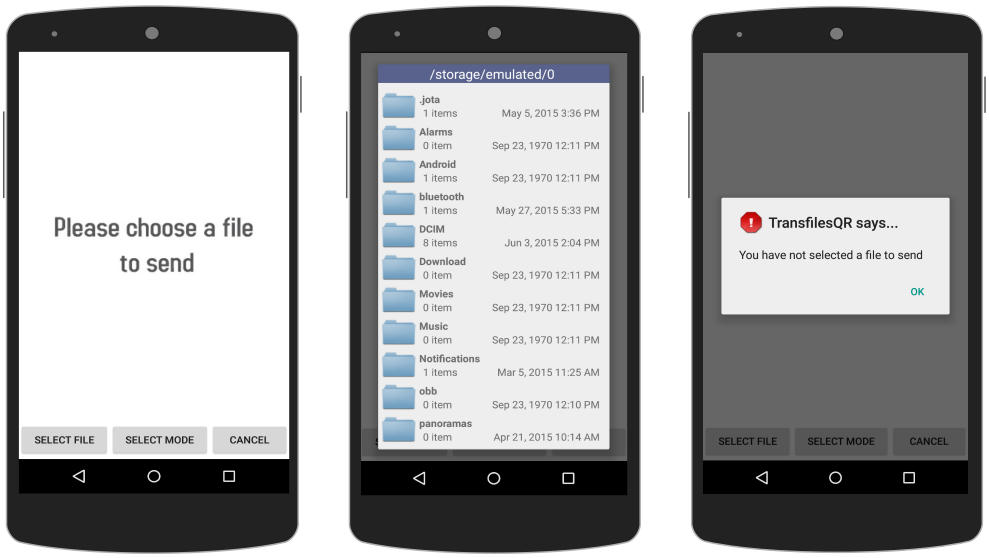
3.3.1 User interface

The user interface design for TransfilesQR uses a few elements to improve the usability of the application and reduce the number of actions (i.e, the number of clicks on the buttons) to perform the file transmission. These elements

comprise: a *Button* for selecting actions to perform, *Dialog* for displaying options and messages to the user and *Toast* for displaying feedback to the user about the status of the transmission. Figures 3.6, 3.7 and 3.8 depict the interface of the application. We hid the camera preview in order to leverage all the space on the display for rendering the barcodes and thus increases the chances to decode a barcode at first try. This approach also allows to use barcodes of more data capacity which means that it can still be seen from longer distances.

Once the application is launched, the main interface provides three buttons: *Select File* for choosing the file to transfer by using a file explorer, *Select mode* for selecting the transmission mode (i.e., sender or receiver) and *Cancel* for cancelling the current transmission.

In order to start the transmission, the user has to choose the file to be transmitted. Figures 3.6(a) and 3.6(b) show the appearance of the main interface and the file explorer, respectively. In addition, application validates the selection of the file before proceeding with the file transmission as shown in Figure 3.6(c).

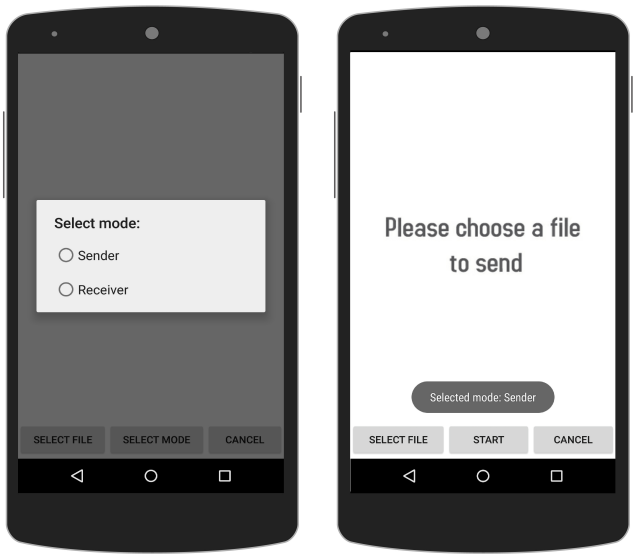


(a) Main interface (b) File explorer (c) Feedback dialog

Figure 3.6: Main interfaces in TransfilesQR

Since the same application is running at both smartphones, the user has to select the mode of transmission and then application notifies to the user the selection mode as shown in Figure 3.7(a) and 3.7(b) respectively. Once the

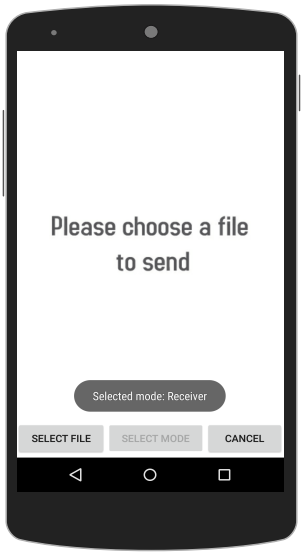
selection mode has been performed, the behaviour of the middle button will change on the sender and the receiver side. In case of the sender, an *Start* button will appear in the application whereas, the button on the receiver side will be disabled and the smartphone will wait for the handshake until the sender clicks on the start button. When the user clicks on the button, the session establishment is performed and eventually, the file transmission. Figure 3.8(a) shows the message when the session establishment starts on the sender side. Figure 3.8(b) and 3.8(c) depict the messages on the receiver side when it is waiting for the session establishment. Finally, when the transmission finishes and the session is closed, both smartphones vibrate for 1 s and a message is displayed on the screen to indicate the total time for file transfer. Figure 3.8(d) depicts the interface when the transmission is over.



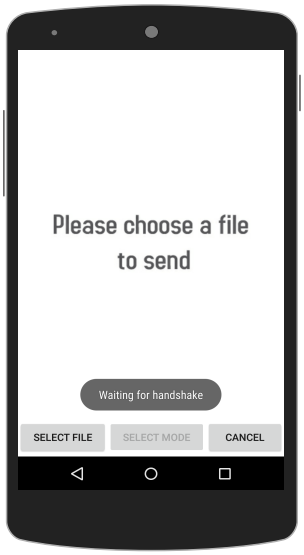
(a) Mode selection (b) Selected mode

Figure 3.7: Interfaces for mode selection in TransfilesQR

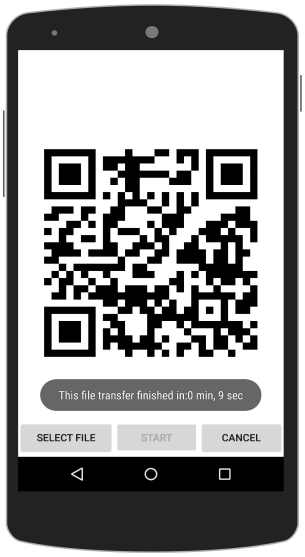
(a) Start handshake



(b) Receiver mode



(c) Waiting for handshake



(d) End of transmission

Figure 3.8: Interfaces for data transmission in TransfilesQR

Chapter 4

Evaluation

In the following, we evaluate our file exchange application by measuring the performance of the proposed protocol, including the proper conditions allowing successful file transfer.

Furthermore, since our solution is a mobile application, we also measured the power consumption of the smartphone because we can obtain valuable information about the resource usage.

This chapter describes the experimental setup used for the evaluation of the application (i.e., the devices used as well as the measurement tools and metrics chosen for evaluation). In addition, it presents the results obtained and provides a brief analysis which are then detailed in Chapter 5.

4.1 Experimental setup

We employed two identical LG Google Nexus 5 smartphones, whose relevant characteristics are summarized in Table 4.1. By using the same smartphones, we ensured that the application is executed under the same conditions, for instance, display resolution, screen size, processor and battery capacity. Additionally, both smartphones had the latest version [1] of Android namely, version 5.0 (Lollipop), which provides a new user interface and includes several features for using the resources of the phone more efficiently [34].

For testing the application, we attached each smartphone to a book holder so as to keep them straight and with the screen facing each other as illustrated in Figure 4.1. In all the experiments, we disabled network connections, set brightness to the middle of its range, and closed all the processes running in background. In this manner, we could better characterize the power consumption of our application [35]. In addition, we performed the experiments

[1] At the time the experiments were performed.

under uncontrolled ambient light conditions to observe how this factor could affect data transmission.

Google Nexus 5		
Software	Operating System	Android 5.0 (Lollipop)
Hardware	CPU GPU	Qualcomm Snapdragon 800, 2.26 GHz processor Adreno 330, 450 MHz
Display	Resolution Type Size Pixels per inch (PPI)	1920x1080 pixels Full HD IPS 4,95 inches 445
Camera	Rear camera Front camera	8 megapixels 1.3 megapixels
Wireless	Bluetooth	Low Energy 4.0

Table 4.1: Device specifications [1]

Figure 4.1: Experimental setup

We considered the following scenario for the measurements. On the one hand, another user launches the application and selects a file to transfer to

the other side. Then, the user selects the *Sender* mode and wait until the other side is available to transfer. On the other hand, a user will do the same steps previously described except for selecting the *Receiver* mode. The measurements for power consumption cover not only the duration of the file transfer, but also the sequence of steps to start such a transfer. For the other measurements, we only consider the time from the moment that session is established until the file transfer finishes and the session is closed. It is important to mention that all measurements have been taken at the receiver side.

There are several options available for measuring power consumption: applications installed on the phone itself or additional hardware that is connected to the phone. In our case, we used Trepn profiler [36], which provides a set of different statistics (e.g., CPU load, GPU load, CPU frequency, GPU frequency, etc.) in order to analyse and detect possible issues with power consumption. Additionally, we performed the same experiment using BLE (Bluetooth Low Energy) for the file transmission. Our goal was to compare these two solutions and observe their behaviour during transmission.

For measuring power consumption, we performed 5 iterations of the experiment; as the obtained result were similar, we show the result of only one iteration in the following. For the rest of experiments, instead, we performed 30 iterations and calculated the average values for analysing the results.

4.1.1 Metrics

We have chosen different metrics to evaluate the application not only in terms of communication performance, but also in terms of power consumption. Specifically, we have chosen the following metrics.

- **Throughput** obtained as the file size divided by the total transfer time.
- **Average power consumption** collected by the profiler during the file transfer between the smartphones.
- **CPU load** as the percentage of CPU (Central Processing Unit) usage as a function of the time.
- **GPU load** as the percentage of the GPU (Graphic Processing Unit) usage as a function of the time.

4.2 Experimental results

In this section, we present the results of our measurements followed by an analysis which will be further extended in Chapter 5. We have divided this section in two parts: the first describes the result about the performance of the file transmission whereas the latter focuses on power consumption.

4.2.1 Protocol performance

Our first experiment consisted of finding the maximum distance that application can support for data transmission. We tested the application with a file of 1.04 Kb and measured the total time for file exchange in a range from 17 cm up to 30 cm as shown in Figure 4.2. Table 4.2 provides the times obtained for each distance. It can clearly be seen from the results that increasing the distance also increases the time (of about one second) in most of the cases. The shortest transmission was reached at a distance of 17 cm with a total time of 7.99 seconds whereas, the highest value was reached at a distance of 30 cm with a total value of 9.97 seconds.

From Figure 4.2 we could see that results seem to follow a linear trend, but there is some dispersion that we associate to the fact that barcodes are not always decoded at first try which causes an increase in the total time for the file transmission. During the experiment, we tested other ranges of distances such as minor than 17 cm and greater than 30 cm but, we noticed that session establishment became more difficult between the phones, resulting in unsuccessful transmissions.

Distance (cm)	**Average time (s)**
17	7.99
20	8.72
25	8.81
30	9.97

Table 4.2: Average time for file transmission as a function of the distance

At the end of this experiment, we chose 17 cm as the distance for performing the other experiments because the values obtained were more precise and there was less dispersion among them.

After fixing the distance used for the rest of experiments, we evaluated the impact of using different barcode sizes. To this end, we changed the frame size by increasing the number of bytes allocated to the payload. In

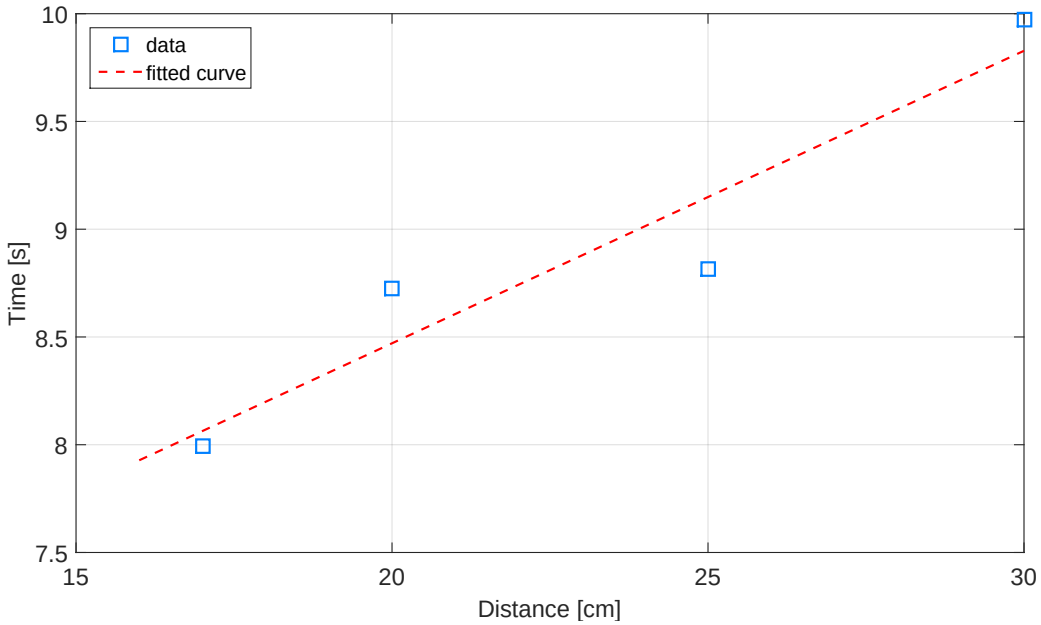


Figure 4.2: Average time as a function of the distance between smartphones

the case of the header, we kept it a fixed to 13 bytes. Figure 4.3 depicts the average time of data transmission based on the barcode capacity. It is of note that the overall trend for the measurements showed a decrease in the time when barcode size increases. Table 4.3 presents the numeric results to support this trend. Indeed, increasing the barcode size also reduces the number of frames to show on the display, therefore, the rendering of frames gets reduced. The total time for transmission is affected considerably when using a barcode of a small size. This fact is apparent when using a payload of 20 and 40 bytes, the total time reached values of 20.04 seconds and 11.90 seconds respectively. Moreover, we noticed a slight difference for barcode sizes of 60 and 80 bytes with a difference of only 0.06 seconds. We relate this result with the continuous processing tasks of CPU which could influence the time when the payload is large.

Similarly, we evaluated the file exchange of different sizes between the smartphones in order to calculate the throughput reached and the time it takes to transfer those files. Table 4.5 presents the results for this experiment. We tested the transmission of text files with sizes between 1.04 Kb and 2.50 Kb with a frame size of 93 bytes. The total time reached values between 7.99 seconds and 15.63 seconds. The increase in time is expected since the number of frames increases which means that more frames will be displayed on the

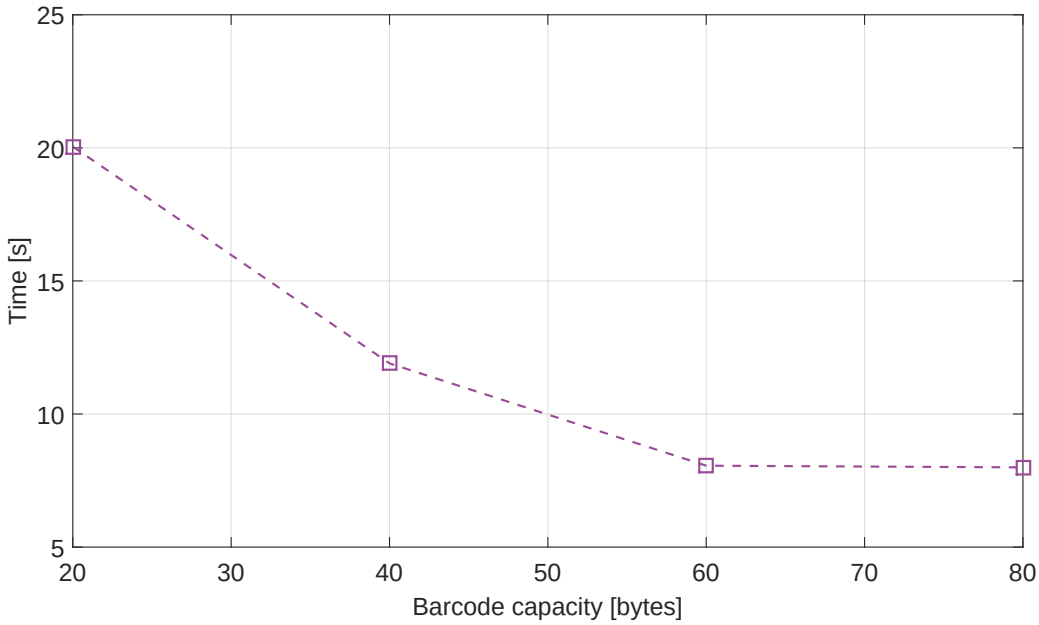


Figure 4.3: Barcode capacity versus time

Frames	Payload (bytes)	Time (s)
54	20	20.04
27	40	11.90
18	60	8.05
14	80	7.99

Table 4.3: Average time for file transmission as a function of the barcode capacity

phone's display therefore, it will take time until sender (receiver) captures all the frames. Figure 4.4 depicts the results of time when increasing the file size. In addition, we tested the exchange of an image that is represented in the last point (9.59 Kb). The time for exchanging this file substantially increases by reaching an average value greater than one minute.

Overall, the results shows a linear trend as expected. Besides, the points obtained (squares) are aligned close to the line which indicates that there is not a meaningful dispersion among the values.

With the same files we calculated the throughput defined as the file size divided by total time of the transmission. Figure 4.5 illustrates the average throughput for the files. From the table above we can see that the throughput

# Frames	File size (Kb)	Time (s)	Throughput (bps)
14	1.04	7.99	130.07
22	1.64	12.13	135.11
27	2.10	12.64	166.03
32	2.50	15.63	156.74

Table 4.4: Average throughput for different file sizes

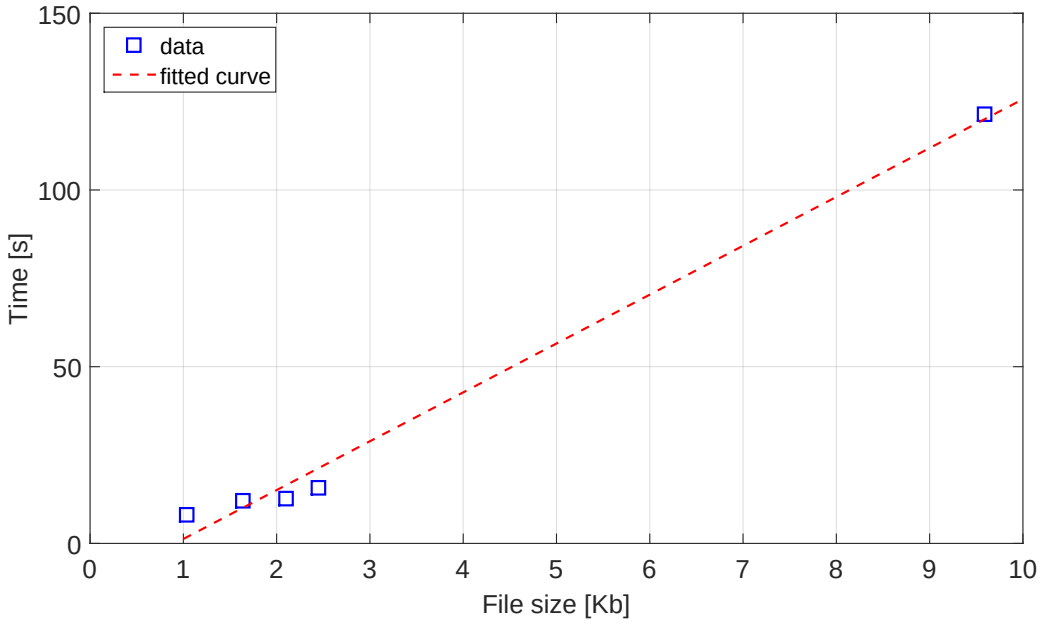


Figure 4.4: Average transfer time as a function of the file size

for the first file (1.04 Kb) starts in 130.07 bps and increases to 135.11 bps for the second file (1.64 Kb). There is a rapid increase in the throughput for the third file (2.10 Kb) with a value of 166.03 bps. When testing the last file (2.50 Kb), we noticed a decrease in the throughput which dropped to a value of 156.74 bps. This change in the throughput could be caused by limitations in the computing capacity of the smartphone as well as, external factors such as lighting conditions that could affect the total time during measurements.

4.2.2 Power consumption

Since our solution is a mobile application, we also analysed the usage of resources during file exchange. In this experiment, both smartphones transfer

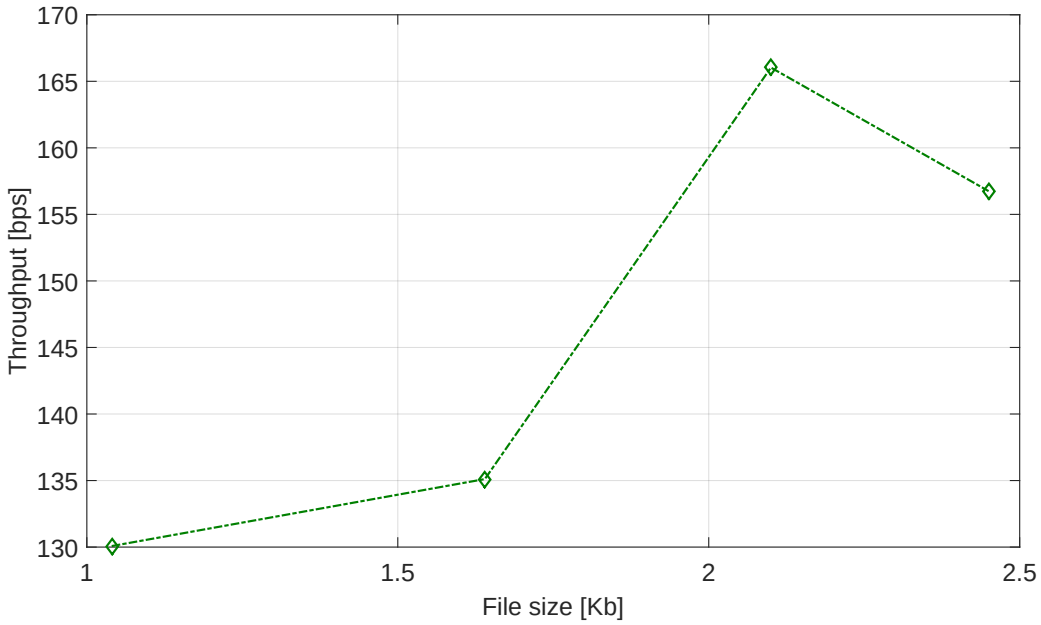


Figure 4.5: Average throughput for different file sizes

files of the same size (1.65Kb). The total time for receiving the file was of about 25 seconds.

First, we measured the CPU load during file exchange between devices. The smartphone used for the experiment has a processor with 4 cores, therefore, we analysed the load on each one as shown in Figure 4.6. It can be seen that until 20 seconds, the CPU load fluctuates between 10% and 60% for three of the four CPUs with one exception for CPU 2 with a value of 90%. We associate these results with the task of searching the file inside many directories and subdirectories, which requires to iterate for each directory and at the same time show the user interface to list the files contained on these directories. Between 20 seconds and 50 seconds approximately, the CPU load varies between 60% and 90%. During the first 25 seconds of this range the transmission is being performed, which means that frames are sent and received simultaneously. The last 5 seconds represent the end of the transmission when the camera is still open and the application is closed. These results are expected since the application is using the camera and also it requires a lot of processing for executing tasks such as showing the current frame on the interface, decoding frames, storing frames received, and notifying the other side about frame received. In addition, this CPU load could increase because decoding barcodes is not always successful at first try.

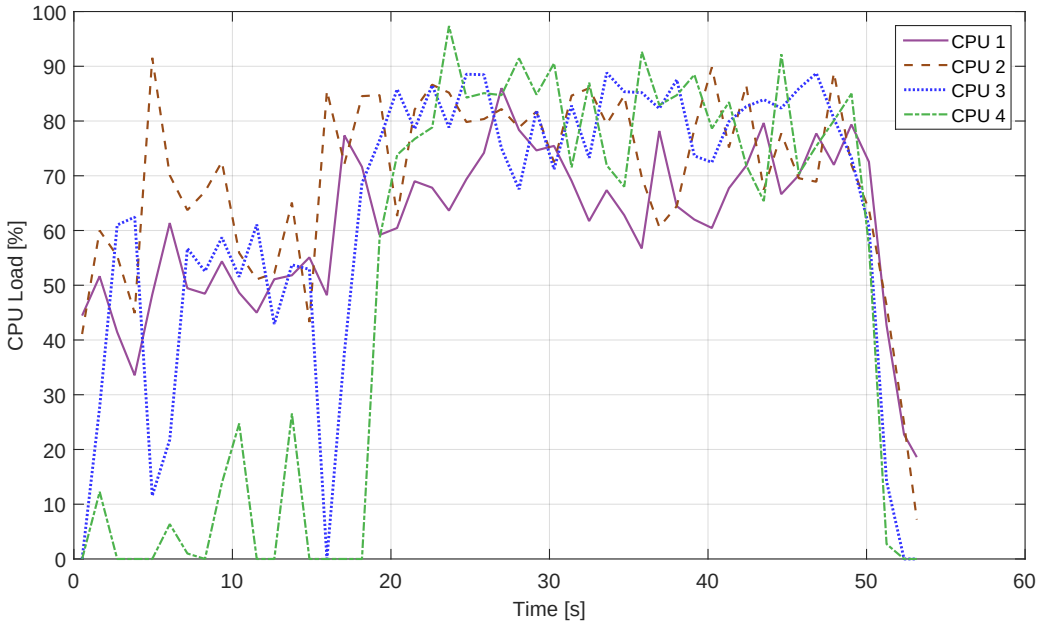


Figure 4.6: CPU load during data transmission

On the other hand, we measured the GPU load on the smartphone. The GPU or Graphics Processing Unit plays an important role in an smartphone since it supports the CPU by processing graphics shown on the display. Thus, we evaluated the contribution of GPU in our application. Figure 4.7 shows the GPU load throughout the execution of the application. We can see that GPU load appears at certain ranges of time. Five peaks can be identified from the figure. The first two peaks appears between 10 to 20 seconds with values fluctuating between 20% and 75%. We associate these peaks with the actions of clicking on the buttons and showing the file explorer with all the file stored in the phone. The peak at 20 seconds reached a value of 40% corresponds the moments that synchronization occurs between the devices. In this point the texture is updated to show a barcode. The last two peaks presents a lower percentage in the use of GPU with value between 8% and 20%. These peaks represent the moment that we close the application, which requires the interaction again with the screen and the buttons.

According to these results, we can see that the GPU is not active during data transmission (range 25-45 seconds) which means that these tasks are completely performed by the CPU.

Finally, the average power consumption measured for the application is shown in Figure 4.8. The obtained results show that application reaches

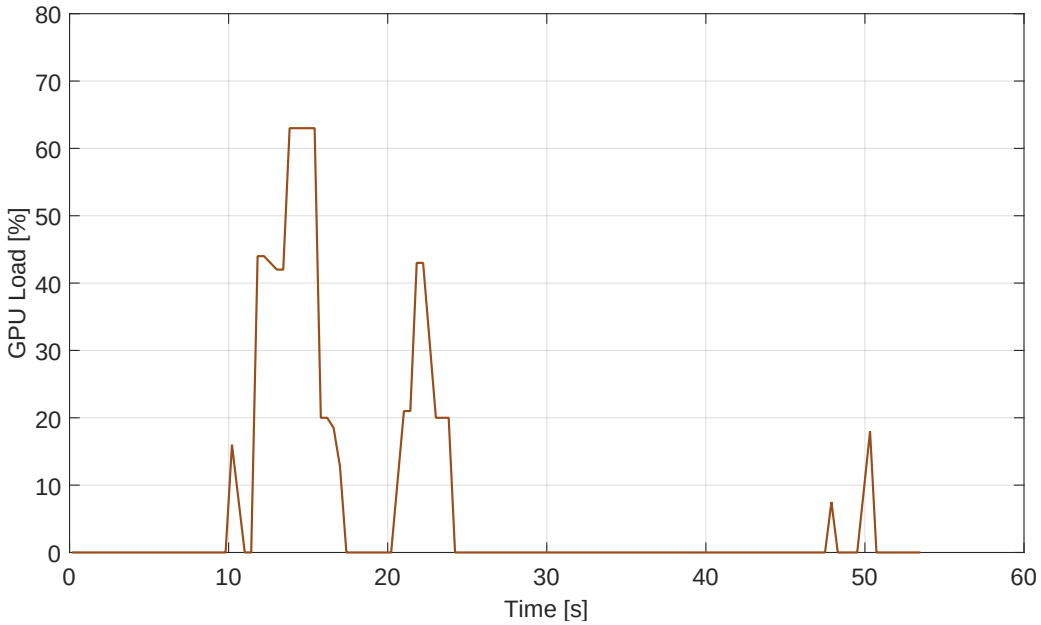


Figure 4.7: GPU load during data transmission

values for power consumption between 2 and 4 W. During the first 20 seconds the application consumes less energy with values between 2 and 3.5 W which correspond to initial steps before performing the data transmission. After 20 seconds, the power consumption increases between 3.5 and 4.4 W. The power draw increase at this point due to the CPU processing while the application is transferring and receiving the file. After 45 seconds, the power consumption decreases which indicates that transmission is over.

In overall, measurements reported a power consumption of 3.4 W, CPU load of 60% and a GPU load of 10%, which leds us to reduce more CPU load and power consumption for our application. Certainly, using hardware such as the camera and the display of the smartphone still represents a high usage of CPU because of the encoding/decoding tasks performed.

As previously mentioned, we measured power consumption of Bluetooth LE as shown in Figure 4.9. The experiment used for the file exchange was a file of 1.65Kb. The profiler collected data from the moment of enabling the Bluetooth on the phone until when it is disabled. It is noticeable, that the time to transfer the file increases more Bluetooth involves different phases before starting transmission. These phases comprises enabling the Bluetooth connection, discovering nearby devices and pairing devices [37]. The total time for file transmission was around 90 seconds. For the first 30 seconds, the

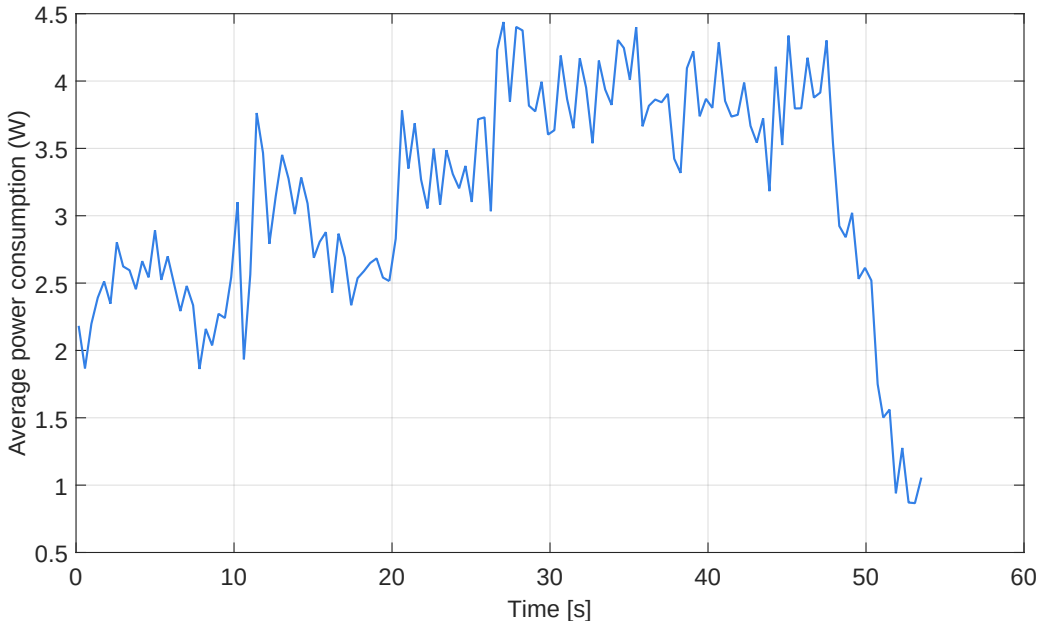


Figure 4.8: Average power consumption during data transmission

enabling of the Bluetooth and discovery phase takes place. At 34 seconds, the smartphone receives a dialog for pairing. Between 40 until 65 seconds, the selection of the file from the file explorer and the selection of the paired phone for sending the file, take place. The time just to send the file is around 4 seconds. Between 65 and 80 seconds, a dialog appears on the device to confirm a file reception from the other smartphone. From 80 until 90 seconds, the user confirms reception and transmission starts. Finally, between 90 and 120 seconds corresponds to the moment for disabling the Bluetooth on the phone and finishing data collection. The results registered an average of 1 W of power consumption during file transmission which is less than the result obtained by our application by in terms of time the result is the opposite.

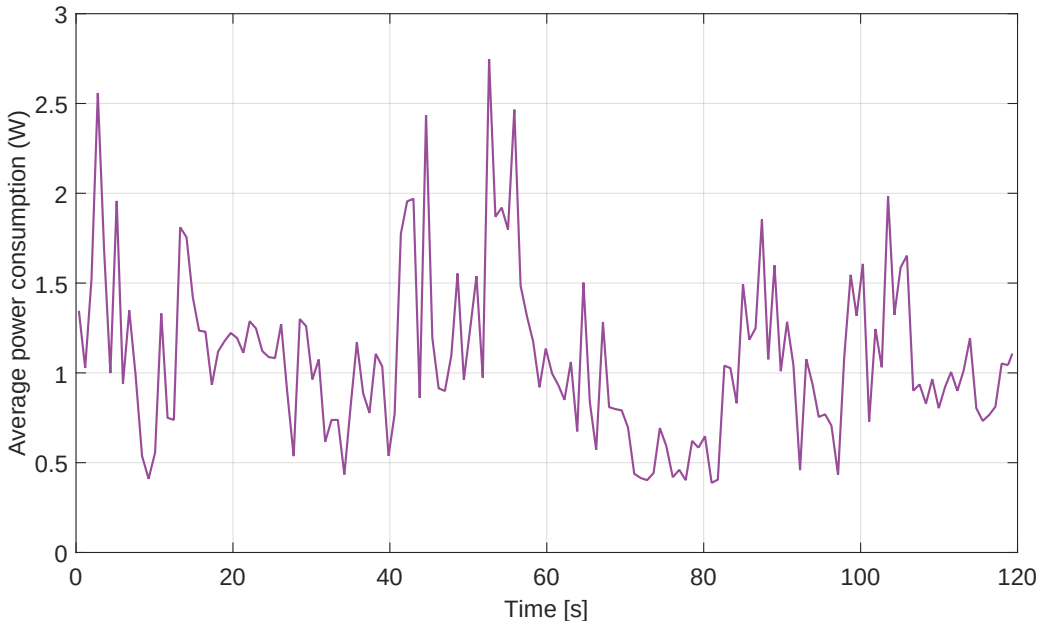


Figure 4.9: Average power consumption using Bluetooth LE for data transmission

Overall, the obtained results show that our application requires the continuous use of hardware components (i.e., camera, CPU, display) during the file transmission, which affects the throughput. In addition, some factors such as lighting conditions and camera resolution contribute to the increase of total time in certain occasions. In terms of power consumption, the application spent more energy than other technologies such as Bluetooth, however in terms of time and usability the application requires less time to transfer the file and less steps to start connection between devices.

Chapter 5

Discussion

We have reviewed the most relevant studies for data communication by using an optical channel. Most of the solutions only present a communication in one direction but still provided an overall idea about the factors that need to be considered for a bidirectional communication. Based on these ideas, we designed a frame transfer protocol and developed a proof-of-concept application. We finally evaluated the performance of our application in terms of throughput and power consumption.

In this chapter, we discuss the related findings, and the factors which affect the performance of our application from different perspectives. They comprise: power drawn, total time for transmission and usability.

The development of mobile applications involves other aspects which are not usually considered when developing web or desktop applications. Particularly, power consumption is paramount for mobile applications due to the fact that battery of smartphones drains out really fast. Therefore, it is important that applications try to make a good use of the resources in the smartphone. These resources mainly refer to the usage of CPU, GPU, memory, display, camera and son on. In addition, the use of other communication technologies for instance, Bluetooth, WiFi and 3G incurs in power consumption.

Some factors to consider in the analysis of power consumption are: the content displayed on the screen (e.g., colors), the level of brightness used, the camera resolution and the type of display of the smartphone.

There are studies revealing that there is a difference in the power consumption when using a white or black background. For a black background the values can fluctuate between 63 mW and 256 mW whereas a white background increases dramatically the consumption reaching values between 197 mW and 527 mW [38]. The obtained value will depend on the color intensity used. In addition, updating the content of the display also contributes to

have higher values for power consumption [39, 40].

The type of display is another factor to consider in the analysis of power draw. There are two type of displays used in smartphones namely, AMOLED (Active-Matrix Organic Light Emitting Diode) and IPS (In Plane Switching LCD Display) displays. The first is a display recently used in new Samsung smartphones which promises to provide better image quality (i.e., high contrast) and the most important is being most efficient in the use of energy. The latter is a display commonly used in LG smartphones which is focused on the enhancement of view angles and color uniformity however, it consumes more power since it requires a backlight [41]. In [42], these two displays were compared and the results shows that not only the fact of using AMOLED displays guarantees a lower power draw because other aspects such as the lighting conditions and the size of the screen could cause the opposite effect.

Similarly, camera's smartphone is a hardware component with a high power cost. Using the camera involves selecting a resolution for the video capture. A high resolution provides a better quality while, at the same time, increasing the power consumption. Not only the resolution incurs a high cost but, also just having the camera on without recording contributes to a high power draw [43].

Since we measured power consumption when using Bluetooth, some aspects will also be discussed as part of our analysis. Bluetooth was created as a wireless technology for short-range communication. Currently, there are two versions of Bluetooth namely, classic Bluetooth and BLE (Bluetooth Low Energy). The first provides high throughput values and a large range of operation whereas the latter is focused in low power consumption and low latency [44, 45]. Based on their specification using Bluetooth requires to perform several steps on both devices (i.e., discovery, pairing). Pairing devices could not be a straight-forward in some occasions since the devices cannot be connected at first try, for instance, the exchange of keys for pairing has failed or hardware or software incompatibility between devices may be experienced. In addition, it is not possible to exchange files simultaneously, which means that a transmission needs to finish on one side first before another can start at the other side.

Based on the above-mentioned considerations, we have enough arguments to explain the results obtained by the application. In terms of hardware specification, the device used has an IPS display, a front-facing camera of 1.3 megapixels and a screen size of 4.95 inches. The interface of the application comprises a white background and the display is continuously updated throughout the data transmission. Considering these facts, it is not surprising to obtain high values for power draw because the camera is always on for getting new images from the preview mode, and at the same time, the

phone is performing encoding and decoding tasks. The use of the IPS display together with the constant update of the display and the colors used by the interface could also be additional reasons of a high power draw. New experiments are required to confirm that changing the approach for rendering frames, improves the use of resources.

On the other hand, the power consumed by Bluetooth resulted to be smaller than that obtained by our application, however in case of the total transfer time, the results were all the opposite. The increase in the transfer time is due to the several steps involved. In our experiment, we had to enable Bluetooth on the phone, wait until the discovery process is complete, open the file explorer to select the file to send, return to the settings window and choose the device to receive the file. Finally, we had to send the invitation to the other device and wait until other party agrees for starting the transmission. As it can be noted, these actions involve many clicks on buttons and the opening of different applications (e.g., settings window and file explorer). In our case, the application does not require to enable any option in the settings of the smartphone since all the features are integrated in one application. Another advantage of our solution is that by using an optical channel for communication is less prone to security attacks.

Moreover, the evaluation of the application showed us that in spite of using a different communication channel, there are other aspects which affect the performance of the communication. These aspects are mostly related to the limitations in the hardware, which are reflected in the values obtained for the power consumption. We also noticed that changing the size of the frames makes a difference in the transmission, which can be leveraged in future work. Since this is the first version of our proof-of-concept, these results provides ideas that can be useful for an optimized version where power consumption is one of the first aspects to improve.

We still need to perform further measurements to observe other factors such as level of brightness used during file transmission, changing the ECC level used in the barcodes, rotations of the smartphone to determine how decoding process is affected in terms of time. It is important to mention that the obtained results cannot be generalized since smartphones specifications may vary for all the brands. However, our findings are useful for the development of bi-directional VLC systems based on smartphones.

Chapter 6

Conclusion

In this thesis, we have implemented a VLC system to enable bidirectional communication between two smartphones. To this end, our solution leverages two components of the smartphone, namely, the front-facing camera and the display. We have designed a transmission protocol for bi-directional file exchange that ensures a successful transmission. We have also developed an Android application that implements such a protocol.

We have finally evaluated our application under several metrics for measuring performance of the proposed protocol and the power consumption as well as the use of resources. The results show that our application can operate in a range between 17 cm and 30 cm for the transmission of files which size is smaller than 3 Kb. Moreover, the throughput is constrained by the computing resources of the phone since several tasks are performed by CPU (e.g., encoding/decoding and rendering frames). In addition, the continuous use of the camera and display of the phone represents a high power draw to the smartphone therefore, new approaches have to be considered to optimize the resources used in the device [46].

To the best of our knowledge, this thesis work provides the first measurement results of power consumption for a VLC solution implemented in smartphones. These results can be considered another perspective to consider, apart of the external/internal factors that affect communication with barcodes.

Overall, this thesis achieved the goals proposed with the implementation and evaluation of application for bidirectional communication; however, it still requires more work for enhancing the throughput and the power consumption. Based on the experimental results, we present the following directions for future research:

- Use a dynamic rate (fps) for rendering the frames on the display in order to update the screen only when necessary.

- Implement an option in the protocol to generate adaptive frames during the file transmission.
- Implement a feature to detect the proper distance for data transmission by using sensors (e.g., accelerometer, gyroscope).
- Study other alternatives of barcodes for the application.
- Evaluate the application with different devices to observe the protocol performance.
- Explore options to increase the throughput by using a more efficient mechanism for error control.
- Perform new experiments to observe behaviour of application when modifying additional parameters (e.g., ECC level, rotation of the phones, level of brightness).

Bibliography

[1] "Google Nexus 5 specification." `http://www.google.com/intl/en_uk/nexus/5/`. Accessed: May 17, 2015.

[2] T. Hao, R. Zhou, and G. Xing, "Cobra: color barcode streaming for smartphone systems," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 85–98, ACM, 2012.

[3] T. Fath, F. Schubert, and H. Haas, "Wireless data transmission using visual codes," *Photonics Research*, vol. 2, no. 5, pp. 150–160, 2014.

[4] M. Xie, L. Hao, K. Yoshigoe, and J. Bian, "Camtalk: A bidirectional light communications framework for secure communications on smartphones," in *Security and Privacy in Communication Networks*, pp. 35–52, Springer, 2013.

[5] W. Stallings, *Data and computer communications*. Pearson/Prentice Hall, 2007.

[6] J. Haartsen, "The bluetooth radio system," *Personal Communications, IEEE*, vol. 7, pp. 28–36, Feb 2000.

[7] M. Uysal and H. Nouri, "Optical wireless communications an emerging technology," in *Transparent Optical Networks (ICTON), 2014 16th International Conference on*, pp. 1–7, IEEE, 2014.

[8] C. G. Lee, *Visible Light Communication, Advanced Trends in Wireless Communications*, ch. 17, p. 520. InTech, 2011.

[9] H. Kato and K. T. Tan, "First read rate analysis of 2d-barcodes for camera phone applications as a ubiquitous computing tool.," in *TENCON 2007-2007 IEEE Region 10 Conference*, pp. 1–4, IEEE, 2007.

[10] S. Dhawan, "Analogy of promising wireless technologies on different frequencies: Bluetooth, wifi, and wimax," in *Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007. The 2nd International Conference on*, pp. 14–14, Aug 2007.

[11] A. Jovicic, J. Li, and T. Richardson, "Visible light communication: opportunities, challenges and the path to market," *Communications Magazine, IEEE*, vol. 51, pp. 26–32, December 2013.

[12] G. Dodig-Crnkovic, "Scientific methods in computer science," in *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia*, pp. 126–130, 2002.

[13] A. Denso, "Qr code essentials," *Retrieved from http://www. nacs. org/LinkClick. aspx*, 2011.

[14] H. Kato, K. Tan, and D. Chai, *Barcodes for Mobile Devices*. Cambridge University Press, 2010.

[15] J.-C. Chuang, Y.-C. Hu, and H.-J. Ko, "A novel secret sharing technique using qr code," *International Journal of Image Processing (IJIP)*, vol. 4, no. 5, p. 468, 2010.

[16] "Denso Wave Incorporated. History of QR Code." `http://www.qrcode.com/en/history/`. Accessed: Feb 20, 2015.

[17] M. Ebling and R. Cáceres, "Bar codes everywhere you look," *IEEE Pervasive Computing*, no. 2, pp. 4–5, 2010.

[18] "Denso Wave Incorporated. Information capacity and versions of the QR Code." `http://www.qrcode.com/en/about/version.html`. Accessed: Feb 20, 2015.

[19] N. Victor, "Enhancing the data capacity of qr codes by compressing the data before generation," *International Journal of Computer Applications*, vol. 60, no. 20, 2012.

[20] T. J. Soon, "Qr code," *Synthesis Journal*, vol. 2008, pp. 59–78, 2008.

[21] S. Arnon, *Visible light communication*. Cambridge University Press, 2015.

[22] D. O'brien, L. Zeng, H. Le-Minh, G. Faulkner, J. Walewski, and S. Randel, "Visible light communications: Challenges and possibilities," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pp. 1–5, Sept 2008.

[23] S. D. Perli, N. Ahmed, and D. Katabi, "Pixnet: interference-free wireless links using lcd-camera pairs," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pp. 137–148, ACM, 2010.

[24] "OpenGL Library." `https://www.opengl.org/`. Accessed: Jun 14, 2015.

[25] A. Motahari and M. Adjouadi, "Barcode modulation method for data transmission in mobile devices," *Multimedia, IEEE Transactions on*, vol. 17, no. 1, pp. 118–127, 2015.

[26] T. Langlotz and O. Bimber, "Unsynchronized 4d barcodes," in *Advances in Visual Computing*, pp. 363–374, Springer, 2007.

[27] T. Yonezawa, M. Ogawa, Y. Kyono, H. Nozaki, J. Nakazawa, O. Nakamura, and H. Tokuda, "Sensetream: enhancing online live experience with sensor-federated video stream using animated two-dimensional code," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 301–305, ACM, 2014.

[28] N. Briscoe, "Understanding the OSI 7-layer model," *PC Network Advisor*, vol. 120, no. 2, 2000.

[29] "Transport Control Protocol Specification ." `https://tools.ietf.org/html/rfc793`. Accessed: Jun 06, 2015.

[30] "OpenCV 2.4.11.0 documentation." `http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html`. Accessed: Apr 28, 2015.

[31] "Android Developers." `http://developer.android.com/reference/android/hardware/Camera.html`. Accessed : Jun 14, 2015.

[32] "Official ZXing ("Zebra Crossing") project home." `https://github.com/zxing/zxing/`. Accessed: Feb 20, 2015.

[33] "OpenGL ES Library." `http://developer.android.com/guide/topics/graphics/opengl.html`. Accessed : Jun 14, 2015.

[34] "Android - 5.0 Lollipop." `https://www.android.com/versions/lollipop-5-0/`. Accessed : Jun 30, 2015.

[35] "Qualcomm Developer Network." `https://developer.qualcomm.com/forum/qdn-forums/increase-app-performance/trepn-profiler/28335`. Accessed : Jun 30, 2015.

[36] I. Qualcomm Technologies, "Trepn Power Profiler." `https://developer.qualcomm.com/software/trepn-power-profiler`. Accessed : Jun 14, 2015.

[37] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.

[38] G. P. Perrucci, F. H. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pp. 1–6, IEEE, 2011.

[39] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone.," in *USENIX annual technical conference*, vol. 14, 2010.

[40] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff, "What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 267–280, ACM, 2012.

[41] K. J. Kim, E. Park, and S. S. Sundar, "Ips vs. amoled: effects of panel type on smartphone users viewing and reading experience," in *Human Centric Technology and Service in Smart Space*, pp. 77–84, Springer, 2012.

[42] S. V. Rajaraman, M. Siekkinen, and M. A. Hoque, "Energy consumption anatomy of live video streaming from a smartphone," in *Proceedings of the 25th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, 2014.

[43] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is energy consumed in smartphone display applications?," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, (New York, NY, USA), pp. 3:1–3:6, ACM, 2013.

[44] "The Low Energy Technology Behind Bluetooth Smart." `http://www.bluetooth.com/Pages/low-energy-tech-info.aspx`. Accessed : Jun 30, 2015.

[45] N. Gupta, *Inside Bluetooth low energy.* Artech house, 2013.

[46] A. Wang, S. Ma, C. Hu, J. Huai, C. Peng, and G. Shen, "Enhancing reliability to boost the throughput over screen-camera links," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, pp. 41–52, ACM, 2014.

Appendix A

Implementation details

On this section, we present more details about the implementation details. Figure A.1 depicts the class diagram of the *TransfilesQR* application.

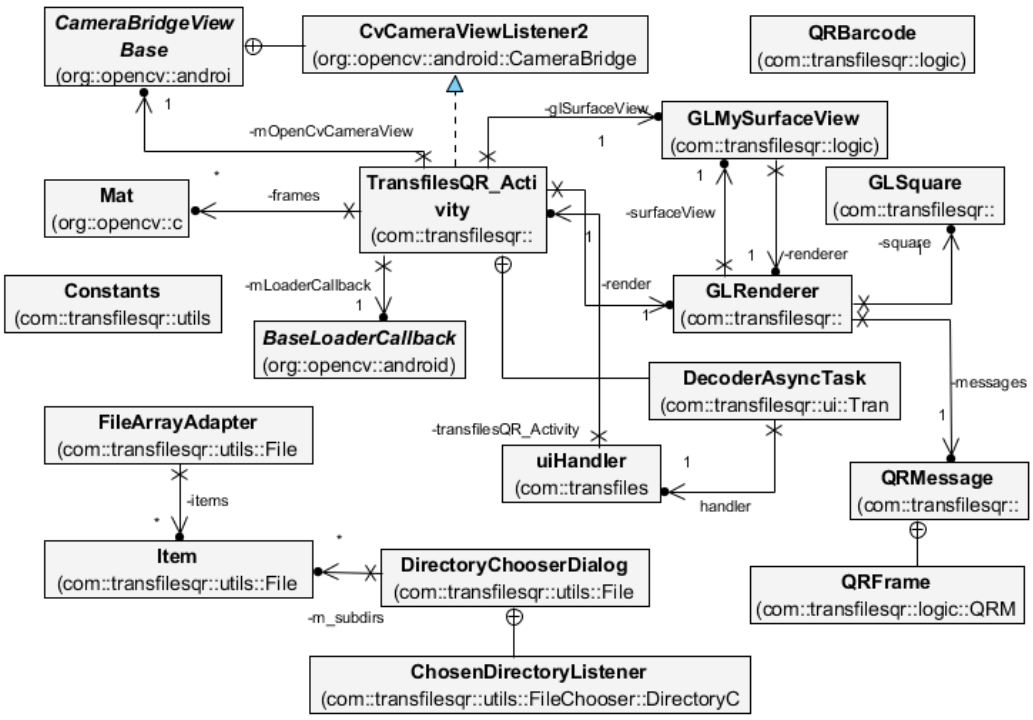


Figure A.1: Class diagram for TransfilesQR

For a better organization of the source code, we have grouped classes into packages according to their use. We defined three packages, namely: ui, for

the user interface; logic, for processing tasks; and utils, for other classes used in the whole application.

The main class in the application is `TransfilesQR_Activity` which is the activity where all the classes and components of the interface are instantiated (e.g., buttons and SurfaceView). At the beginning, the application requires to load OpenCV, therefore, the main activity instantiates the `BaseLoadCallback` class. In addition, three more classes from OpenCV library are required to access the camera of the smartphone namely, `Mat`, `CameraBridgeViewBase`,`CvCameraViewListener2`.

On the other hand, `QRMessage` is in charge of splitting the file into chunks and subsequently, creating the frame based on the protocol specification in the `QRFrame` class. The barcodes are encoded/decoded in `QRBarcode` class where ZXing libraries are invoked. Moreover, the frames are displayed on the screen by using `GLMySurfaceView` and `GLRenderer` classes. To render the frames on the screen, it is required to create first a texture in the `GLSquare` class to place over it.

The control of the frames received/unreceived is performed in the class `DecoderAsyncTask`. Any decision of updating ACK or removing frames already received is performed through the handler `uiHandler`.

The displaying of the file explorer is possible by using the following classes: `FileArrayAdapter` and `Item` are used to obtain the file stored on the device. `DirectoryChooserDialog` and `ChosenDirectoryListener` are in charge of displaying the list of file with an `AlertDialog` and enable the actions to allow the selection of the file.

Finally, `Constants` is a class to store all the constant values used in the whole application. All the settings regarding barcodes and the type of frames used in the protocol are stored in this class.