Delft University of Technology
Master of Science Thesis in Embedded Systems

Designing an optical link between a micro-display and a smartphone camera

Zehang Wu

Embedded Networked Systems

TU Delft Delft University of Technology

Designing an optical link between a micro-display and a smartphone camera

Master of Science Thesis in Embedded Systems

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Zehang Wu
4903269

Wednesday 17$^{\text{th}}$ November, 2021

Author
Zehang Wu (Z.Wu-7@student.tudelft.nl)

Title
Designing an optical link between a micro-display and a smartphone camera

MSc Presentation Date
Thursday 25th November, 2021

Graduation Committee

Dr. Marco Antonio Zúñiga Zamalloa (Chair)	Delft University of Technology
Dr. Asterios Katsifodimos	Delft University of Technology
Miguel Chávez Tapia (Daily advisor)	Delft University of Technology

Abstract

The popularity of various wireless communication applications is crowding the radio spectrum. As an alternative medium, the visible light spectrum is being exploited. Camera-based visible light communication systems are gaining much attention due to the widespread use of smartphones equipped with small embedded cameras. Screens (e.g., liquid-crystal displays) are ideal choices as multi-pixel transmitters that can send high-capacity packets continuously. Although there are many works on screen-to-camera links, only very few are exploiting micro-screens as transmitters. State-of-arts are mainly optimizing the performance of the optical link, less attention is paid to the system's completeness and practical applications. Thus, the goal of this work is to deliver a self-contained visible light communication system that can send information over a micro-screen-to-camera link at a high data rate.

To achieve our goal, an Android application with real-time image reading and processing is developed. Furthermore, a multi-transmitter system is designed to increase the bandwidth of the communication channel and the system is tweaked to transmit information at a high frame rate. The evaluation results show that this work improves the data rate by over 7-fold, from a baseline of 1.5 kbps to 11.4 kbps. Additionally, a standalone prototype is built based on a Raspberry Pi Zero W. Finally, to showcase the potential of the platform, a smart-city application is developed, where users can download information from Google Maps with the newly developed optical link.

Although the result of this project is encouraging, there is much room for improvement. We envision this work will motivate more research near-field, especially on micro-screen-to-camera links.

Preface

This thesis project is carried out as the finalizing part of the Embedded Systems master program at the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Delft University of Technology.

I can still remember the time when I first met professor Marco during a lecture on the course QEES (Quantitative Evaluation of Embedded Systems). I was attracted by the way he gives lively and interesting lectures as well as his patience when helping students to learn. This is why I approached him for a possible thesis topic. I am curious and interested in visible light communication and after a discussion with Marco and Miguel, I decided to work on screen-to-camera communication for my master thesis project. The last 11 months is the toughest period I have ever gone through, not only because of the 'quarantine' working style due to the widespread coronavirus disease, but also the challenges that appear in the project. However, I will never regret that because I learned and grew up a lot, whether in researching, presenting, writing or self-management.

This project is coming to an end, I would like to express my sincere gratitude for those who have helped and guided me along the way. First, I want to thank Marco for giving me this wonderful project and the time you spend guiding and advising me weekly, towards the right direction. It is your patience and guidance which help me get through the first couple of months of this project when I was hardly able to make progress. I would also like to thank Miguel for the instant and detailed replies whenever I have questions, whether they are urgent or not. You always show kindness and helpfulness, from getting your hands dirty to help me get familiar with the hardware setup, to seeking solutions together when I was in trouble and desperate. Special thanks to professor Asterios for being one of the thesis committee members and the understanding when I had to postpone the presentation and defence date. Additionally, I really appreciate the valuable and constructive questions from all the professors and PhDs during the mid-term talk. Last but not least, I would like to thank my parents and grandparents, for supporting me economically and mentally, during this overseas study experience.

Zehang Wu

Delft, The Netherlands
Wednesday 17$^{\text{th}}$ November, 2021

Contents

Chapter 1

Introduction

Wireless communication technologies have fundamentally changed modern social interactions, bringing unprecedented convenience to daily life. By utilizing radio waves instead of physical wires, communication services ranging from interplanetary communications to mobile communication is no longer impractical. Wireless devices are now widely distributed in application areas including but not limited to broadcast radio, Bluetooth connections, mobile cellular networks, local area networks such as Wi-Fi and wireless sensor networks. However, the rising amount of radio spectrum applications and users is leading to spectrum congestion. An alternative to solve this spectrum congestion is to exploit the visible light spectrum. As a subset of optical wireless communication technologies, visible light communication (VLC) utilizes a human-visible portion of the electromagnetic spectrum, which lies in wavelengths from 380 to 780 nanometers. VLC systems send information by modulating lights. For example, a transmitter switches between *on* and *off* states by modulating the intensity of the transmitted light. A receiver can capture this information and decode it as binary `0`s or `1`s. The visible light spectrum has a 10,000 times wider span in spectrum than traditional radio waves, and thus has enormous potential.

Numerous researchers [13, 28, 3, 30, 5, 29, 2, 8] have started to explore VLC systems using single-pixel transmitters (only send one bit at a time) and receivers, such as liquid crystals or light-emitting diodes (LED). To transmit multiple bits at a time, multi-pixel transmitters (e.g., liquid crystal array, LED array) and receivers (e.g., photo-diode array, camera sensors) are studied to a certain extent [12, 4]. Among the multi-pixel transmitters, screens are gaining more and more attention, from monitor-size to smartphone-size [18, 7, 31, 6, 24, 25]. The advantage of using a screen is that more data can be encoded and sent in the form of 2D barcodes (such as QR codes). With the rapid expansion of the mobile phone market, smartphones equipped with cameras are considered an excellent choice as receivers in VLC systems.

This leads to our project, a micro-screen-to-camera VLC system for short-range communication (12 cm). The transmitter in this project is a low power micro-display (5 mm in diagonal) based on ferroelectric liquid crystal over silicon (FLCoS). The micro-screen is extracted from an LV201K display module [15], sold by Control Electronics Co. Similar to Figure 1.1, the micro-screen can modulate incident lights by reflecting or absorbing light and hence shows bright and dark pixels to transmit binary number `0`s and `1`s.

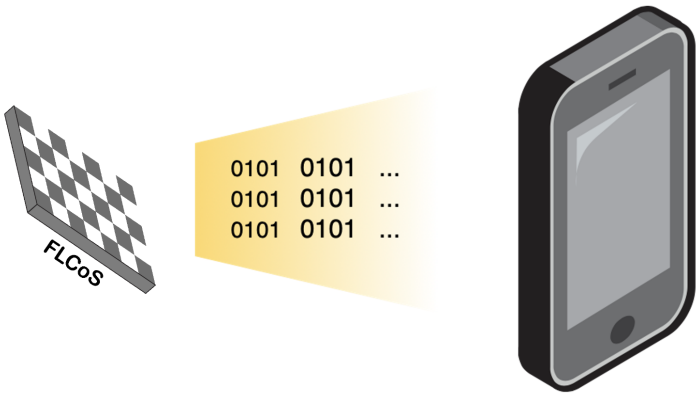


Figure 1.1: **Micro-screen to smartphone camera communication.** Depending on whether a voltage is applied to each liquid crystal cell, the micro-screen reflects or absorbs incident light to show bright and dark pixels and hence transmit binary 0s and 1s.

1.1 Problem statement

Firstly, the default display with a basic implementation provides a low data rate, around 1 kbps using a camera recording a video at 15 frames per second (FPS). There is still considerable room for improvement, whether it is from the software perspective or the hardware view.

Secondly, the micro-screen module requires both a data source and power from an external laptop, which is not ideal for a compact and well-round embedded device. Besides, related works on screen-to-camera links are mainly focusing on optimizing the communication channel instead of building a complete system and a meaningful application.

Considering all of the above aspects, the goals of this project are two. On one hand, to explore ways of data rate enhancement. On the other hand, to prototype a system that operates on its own as well as to demonstrate a potential use case. The research questions are formulated as:

Research question 1: *What are the best approaches to improve the data rate of the micro-screen-to-camera system and which approaches are the most feasible?*

Research question 2: *How can we design and prototype a self-contained embedded VLC system and demonstrate a potential use case?*

1.2 Contributions

Considering the above objectives, the main contributions of this project can be summarized as the following 4 parts:

1. **An Android application that reads images and decodes QR codes in real time.**

When a smartphone is used as a receiver in VLC, the information is normally recorded as a video. A significant amount of time is taken on recording the video file. Moreover, the processing of video frames can only start after the video file is saved and reloaded, which cost an extra time (ranging from hundreds of milliseconds to seconds). This time-consuming process limits the data rate. To eliminate the requirement of recording a video, we developed an Android application that reads and processes image frames in real time. Along with an efficient multi-threading scheme, the processing of images takes advantage of the camera idle times. The real-time implementation of image reading and processing decreases the data *reception time* by 26.7% and increases the throughput and goodput by 46% and 89% compared to the implementation of recording a video.

2. **Deployment of two transmitters to further increase the transmission rate.**
   With only one transmitter, there is a considerable idle portion in the camera preview. To better utilize the camera sensor, two transmitters are deployed, which doubles the channel bandwidth and halves the transmission time. As the baseline setup, the information is sent at 15 FPS and received at twice the frequency (30 FPS) according to the *Nyquist Sampling Theorem*. Thanks to the integrated Reed-Solomon error correction codes, we 'violate' this restriction and increase the transmission rate to 60 FPS. The *transmission time* is cut down by several folds and the data rate reaches up to 11.4 kbps (throughput) and 10.7 kbps (goodput).

3. **Simple prototype design of a self-contained embedded system.**
   The current system cannot function well without an external laptop for both data source and power supply. To achieve a self-contained system, we present a prototype design consisting of 2 Raspberry Pi Zeros, 2 sets of FLCoS micro-display modules, a 2700 mAh lithium ion battery as well as a solar harvesting panel.

4. **An demonstration application that sends locations and shows them on a map.**
   To demonstrate a potential use case, we provide an application that transmits locations and display them intuitively on a map. This involves encoding information with locations and integration of GoogleMap Software Development Kit.

1.3 Organization

The remainder of the thesis is organized as follows: Some important background knowledge and related works are described in Chapter 2. Then, the implementation and system design corresponding to the two goals (improving data rate and self-contained system) are explained in detail (Chapters 3, 4). The subsequent Chapter 5 contains the evaluation results. The conclusion is provided in the last chapter (Chapter 6) as well as potential future works.

Chapter 2

Background and Related Works

To help the reader better understand this project, some necessary background knowledge on light polarization, the working principles of liquid crystal display and some concepts in data transmission are briefly explained in Section 2.1, 2.2, 2.3. Afterwards, Section 2.4 discusses and summarizes the state-of-arts works related to this project.

2.1 Polarization of light

Light and other electromagnetic waves are traverse waves consisting of varying electric and magnetic fields. Both the electric (blue arrows in Figure 2.1) and magnetic fields (red arrows in Figure 2.1) oscillate perpendicular to the direction of propagation. *Polarization* is the phenomenon where traverse waves fluctuate in a definite direction relative to their propagation [19]. The direction of polarization is defined to be the same as the wave's electric field. Light is called *polarized* if the direction of the electric field of light is fixed. If the direction of this electric field fluctuates in all directions, it is called *unpolarized*. Many common light sources such as sunlight, incandescent bulbs and LED generate unpolarized light.

Polarizers are light filters made from polaroid materials, which allow light in only one polarizing direction to pass through. The electric field of unpolarized light is a vector that can be divided into parallel components and perpendicular components to the polarizer's transmission axis. An ideal polarizer passes the parallel component and blocks the perpendicular component. If the angle between the polarization of light and the transmission axis of the polarizer is θ and the amplitude is E_0), then the amplitude of the filtered light is $E = E_0 \cos\theta$. If the intensity of the incident light is I_0, the intensity of light that passes through the polarizer is described as the **Malus' law**:

$$I = I_0 \cos^2\theta, \tag{2.1}$$

Averaging (2.1) over all the angles leads to:

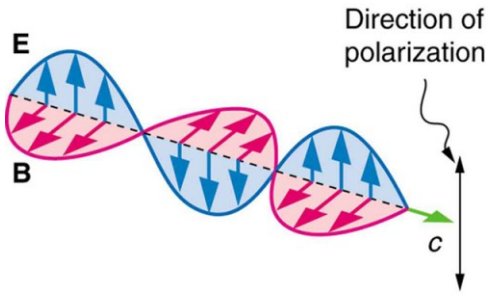
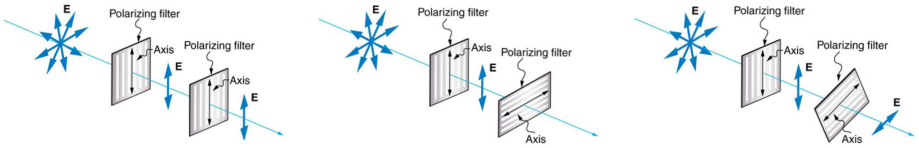


Figure 2.1: **Polarization of light [19].** Light and other electromagnetic waves are traverse waves consisting of varying electric and magnetic fields. The direction of polarization is defined to be the same as the wave's electric field (E).



(a) . (b) . (c) .

Figure 2.2: **Effect of polarizers [19].** By placing the two polarizers such that their transmission angles are parallel, the incident light can pass through. If the transmission angles are perpendicular, light is blocked. Other angles will allow a portion of the light to pass through.

$$I = \frac{I_0}{2}, \tag{2.2}$$

which means in practice, unpolarized light passes through a polarizer will lose at least half of its intensity.

2.2 Liquid-crystal displays (LCD)

The working principle of liquid-crystal display (LCD) technology relies on the polarization of light. Liquid crystal cells are capable of changing or maintaining the polarization direction of the light that passes through them. A typical liquid crystal cell consists of two vertically-placed polarizers and the liquid crystal material in between. The two working states of a liquid crystal cell are illustrated in Figure 2.3. When no voltage is applied on the two electrodes, the polarized light coming out from the first polarizer will be twisted 90 degrees, which is aligned with the second polarizer (named *analyzer*) and hence passes through to the output side. With an applied voltage, the twisted structure is broken. Incoming light maintains the same polarization state and is blocked by the analyzer.

The FLCoS micro-display deployed in this work is a special type of liquid-crystal display, which shares the same operating characteristics as a traditional LCD.

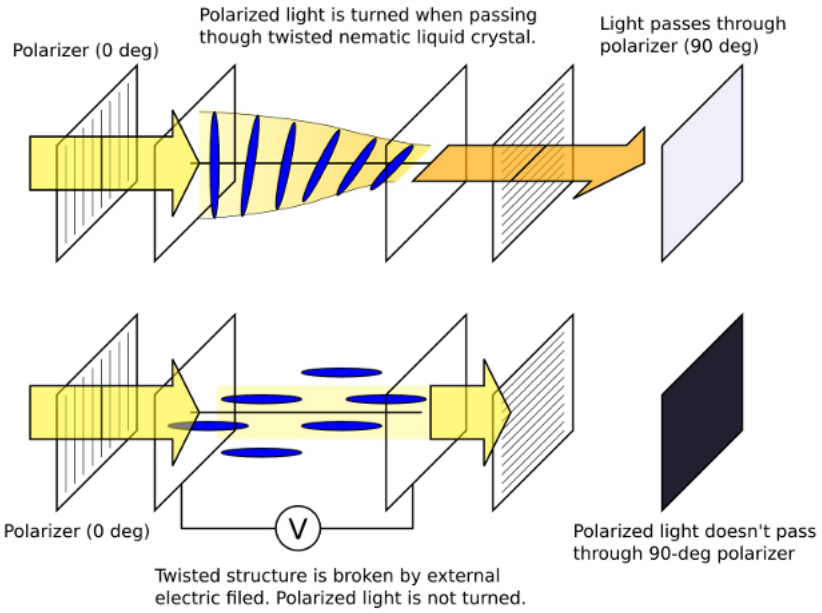


Figure 2.3: **Working principle of liquid crystal [1].** By controlling the voltage applied to the liquid crystal cell to on or off, the polarization of incident light is maintained or rotated 90°, which lead them to be blocked or pass through and show dark and bright pixels.

2.2.1 Transmissive, reflective and transflective LCD

LCDs require an external light source to display contents. Depending on the placement of the light source and how the light travels to the viewer, they are characterized into three types (Figure 2.4):

- *Transmissive* LCDs are embedded with blacklight panels where light passes through the display. It works well indoors but suffers from visibility issues in direct sunlight.
- *Reflective* LCDs required ambient light sources such as sunlight and show contents by modulating and reflecting the incident light.
- *Transflective* LCDs are the combination of the above two. It has an embedded backlight to work in dark environments and a reflective component for bright environments.

The FLCoS micro-display module in this work is reflective, which uses a silicon backplane to reflect light. A detailed description is provided in Appendix A.

2.3 (Wireless) data transmission

2.3.1 Nyquist-Shannon sampling theorem

In signal processing, sampling is the process of recording a continuous signal into discrete values. The sampling frequency should be high enough to prop-

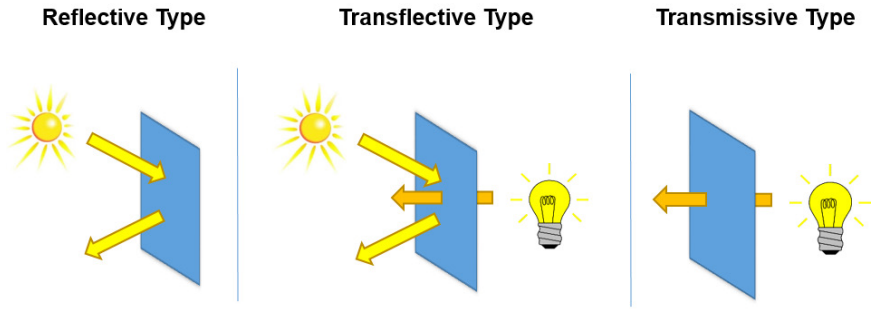
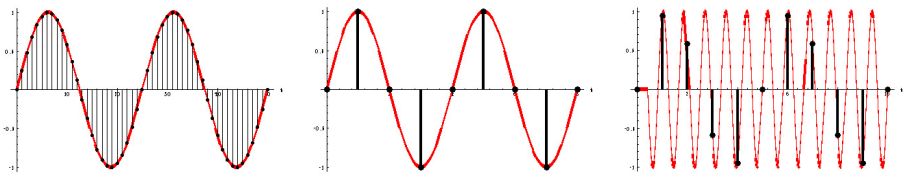


Figure 2.4: **Different LCD types [26].** Reflective LCD absorbs or backscatter light to show contents. Transmissive LCD block or allow light to pass through to show contents. Transflective LCD is a combination of the previous two.



(a) **Over-sampled.** (b) **Well-sampled.** (c) **Under-sampled (Aliasing).**

Figure 2.5: **Sampling in different frequencies [21].** For a sinusoidal signal, sampling at a frequency that is over twice the source is good but inefficient. A sampling at exactly twice the source frequency can capture the essence. A sampling frequency lower than twice the source frequency will cause aliasing.

erly record the source information, otherwise, it leads to aliasing (incorrect re-construction of the original information). The Nyquist-Shannon sampling theorem states that for a discrete sequence of samples to capture all essential information from a continuous signal, the sampling rate should be at least twice as high as the signal frequency. For example in Figure 2.5, sampling a sinusoidal signal with over twice its frequency is more than enough to reconstruct the signal, but it is inefficient since it will generate a large amount of data that might not be useful. When sampling with less than twice the frequency of the sinusoidal, the samples captured are not sufficient to represent the original signal, which leads to *aliasing*. Exactly at twice the frequency of the source signal, the samples capture the essence.

For capturing images, the Nyquist-Shannon sampling theorem is applied not only to time but also to space. Spatial frequency refers to the structure size in sampling (i.e., resolution). A smaller structure size means higher spatial frequency. For capturing two-dimensional QR codes, the pixel size of camera capture should be at least half of the smallest module size in the QR code.

Based on the Nyquist Sampling Theorem, the baseline frame rates of the transmitter and the receiver in our work are configured to 15 FPS and 30 FPS respectively.

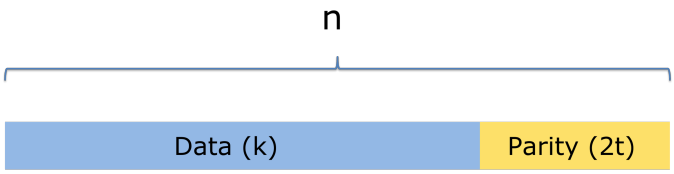


Figure 2.6: **Layout of Reed-Solomon error correction code.** A Reed-Solomon code $RS(n, k)$ (with total n symbols) consists of k data symbols and $n - k = 2t$ parity symbols is able to recover from t error symbols with unknown location or $2t$ erasures (error symbols with known locations).

2.3.2 Reed-Solomon forward error correction

Transmission errors are unavoidable in communication links especially wireless links. Two basic strategies for dealing with errors are **error correction** and **error detection**. For error correction, enough redundant information is added to the data such that the receiver can recover the original data regardless of the allowable data loss. On the other hand, error detection only includes enough redundancy to allow the receiver to know whether an error has occurred (but not which error) and have it request a re-transmission. Our VLC system is a simplex communication system, the receiver cannot request a re-transmission if it detects an error. Hence error correction codes are applied to cope with transmission errors.

Reed-Solomon error correction is commonly applied in data transmission and storage for its strong error-correction properties. It is based on the fact that every n degree polynomial is uniquely determined by $n+1$ points. Extra points are redundant but useful for error correction. An example of how it works is [22]: A first-degree line is determined by two points and they are sent with two extra points that also lies in this line. If one of the points results in an error during the transmission, the original line is still able to be reconstructed since three of the points are in the same line while the error one is not.

Reed-Solomon code consists of linear s-bit code blocks, which are named *symbols*. As in Figure 2.6, a Reed-Solomon code is specified as $RS(n, k)$. This means the whole message (or *codeword*) is constructed with k data symbols of s bits and $n - k = 2t$ parity symbols. When $2t$ packets (symbols) of redundant information is added to the transmitted data, a Reed-Solomon code is able to correct up to t error packets. For example, a 255-packet message with 64 packets of redundancy included means 32 error packets are correctable. If the location of error packets are known, they are marked as *erasures*. In this case, the correction code is able to recover $2t$ error packets (64 packets in the example).

Because there are missed packets during transmission, the message is integrated with Reed-Solomon error correction codes to increase the reliability of the system. A detailed explanation of the message composition is provided in Section 3.1.1.

2.4 Related works

2.4.1 Single-pixel transmitter

Many researchers explored VLC systems with a single-pixel transmitter such as a light-emitting diode (LED) or a liquid crystal (LC) cell.

LC/LED-to-photodiode

RetroVLC [13] is a bi-directional VLC system that uses a LED lamp attached with a photodiode as a light source and a data receiver. The transmitter is a credit-card-size tag device consisting of an LCD shutter, a photodiode and a solar cell. The LC shutter can backscatter and modulate the light for data transmission using On-Off Keying (OOK) and Manchester coding. It achieves an uplink (LC-to-photodiode) data rate of 0.5 kbps and a downlink (LED-to-photodiode) data rate of 10 kbps at up to a 2.4m distance. *PassiveVLC* [28] is a more comprehensive design and implementation of *RetroVLC* that improves the uplink data rate (LC-to-photodiode) to 1 kbps with an optimized trend-based modulation. *Luxlink* [3] is a passive LC-to-phototransistor VLC system that does not rely on an active light source but ambient light. A 6x8 cm LCD is deployed and the Frequency-shift keying (FSK) modulation scheme is adopted to achieve a data rate of 80 bps at a distance of 4 meters indoors and 60 meters outdoors.

LC/LED-to-camera

Instead of a photodiode, other works use smartphone cameras as receivers. To address the flickering issue when using a camera as a receiver, *Pixel* [30] utilizes the polarization property of the liquid crystal. However, the data rate is only 14 bps.

Many works pair a LED with a smartphone camera for communication. [5] introduces a bi-directional VLC system between an LED and a smartphone. It transmits data at 2 kbps from the LED to the smartphone's camera and 30 bps from the smartphone's flashlight to the LED. *InfoLED* [29] and *LightAnchors* [2] build augmented reality applications through LED-to-camera links by modifying existing indicator LEDs in electronic devices. Without breaking the original indicator purposes, they encode and transmit data by flickering the LED at high frequencies. *ColorBars* [8] is another LED-to-camera communication system that explores the RGB color channels. By encoding data into different colors of the LED, the data throughput reaches 5.2 kbps with Google Nexus 5 as a receiver.

Discussion

The above studies use an LED or a liquid crystal as a transmitter while this work utilizes a FLCoS (ferroelectric liquid crystal on silicon) micro-display. However, the micro-screen we used is also based on the polarization of light and liquid-crystal shutters (Appendix A explains the working principle of the FLCoS micro-display). A VLC system with a single-pixel transmitter typically has a long operation range (from several meters up to 60 meters) since the receiver only need to capture the light of a single point (pixel). A limitation of these systems

is that they can only send one bit at a time. A multi-pixel transmitter would be the solution to this limitation but with a shorter operation distance because more information needs to be captured for each frame. Our goal is to achieve a high data rate for a short-range communication system. Thus, rather than using a single-pixel transmitter like an LED or liquid crystal, this work uses a FLCoS micro-screen, which is a multi-pixel transmitter.

2.4.2 Multi-pixel transmitter

To send more than one bit at a time, some other works use multi-pixel transmitters such as LED arrays. *Luxapose* [12] achieves accurate (decimeter location error and 3° orientation error) indoor positioning with an array of LED landmarks and smartphones. [4] utilizes an 8x8 LED array for data transmission and smartphones can receive data at 1.25 kbps.

Apart from the multi-pixel transmitters that simply deploy an array or matrix of single-pixel transmitters (LEDs in the above cases), screens are gaining more and more attention because of their high-resolution characteristics. Screens enable more advanced modulation schemes. Typical usage is encoding data into 2D barcodes (such as QR codes) for transmission. Related screen-based works utilize screens in various sizes, from monitor-size to micro-screens.

Large screen: computer monitor

Pixnet [18] is one of the earliest screen-to-camera VLC systems. It utilizes a 30-inch monitor as transmitter and DSLR cameras (Casio EX-F1 and Nikon D3X) as receivers to set up a simplex communication system. On the transmitter side, they fill the screen with as many data symbols (in the forms of QR codes or custom OFDM symbols) as possible. The cameras are recording at a frame rate of 60 FPS. The data rate can reach up to 12 Mbps at a distance of 10 meters. *Pixnet* also works in view angles as wide as 120° but with a lower data rate (8 Mbps). They also experimented with a smartphone (Nokia N82) at a distance of 2 meters, which attains up to a data rate of 4.2 Mbps. *Focus* [7] surpasses *Pixnet* with an improved OFDM-based visual code design that encodes data in many independent sub-channels in the frequency domain. *Focus* uses Android phones (Galaxy S6 and Nexus One) for evaluation. The display frame rate of the screen is set to half of the readers' capture rate and is separated away from a distance between 35 and 200 cm. For all distances in their evaluation, the throughput obtained using Galaxy S6 is over twice *Pixnet*'s, up to 240 kbps. With a high-end DSLR camera (Nikon D7100) recording at 60 FPS, *Focus* can obtain data rates as high as 2 Mbps.

ChromaCode [31] represents an emerging paradigm in screen-to-camera communication that embeds data imperceptibly into regular videos. It achieves data transmission at a remarkable 700 kbps throughput and 120 kbps goodput while being unobtrusive to human eyes. The transmitter in *ChromaCode* system is a 27-inch monitor with a 120 Hz refresh rate and the receiver is an Android smartphone.

Small screen: smartphones' screen

Previously described works use large-size (e.g., 33 inch) screens as transmitters, *Cobra* [6] presents a 2D color barcode design for small-size screens (smartphones). The highlight of *Cobra* is that it adapts the size and layout of the code blocks in the streamed barcodes to deal with image blur. An evaluation between the screen and camera of smartphones (Google Nexus S and HTC Inspire) is conducted, in which *Cobra* can reach a data rate up to 172 kbps. *RDCode* [24], increases the throughput of *Cobra* by enhancing the transmission reliability. What they contribute are error correction schemes at three levels (intra-blocks, inter-blocks nad inter frames) for recovering lost blocks and frames. In their evaluation, *RDCode* achieves a throughput of 174.4 kbps, which is over two-folds of *Cobra*'s (70.4 kbps) under the same condition. Similar to *RDCode*, *RainBar* [25] also improves the design of the color barcode. Rather than error correction, *RainBar* present a high-capacity barcode layout design. For the same size of a screen, *RainBar* has 2.5 more columns and 4 more rows blocks in the barcode than that of *Cobra*. This enhancement is verified in their experiments where the maximum throughput of *RainBar* is 955.68 kbps than *Cobra*'s 518.09 kbps under the same test environments.

Micro-screen

As far as we know, there are very few works in screen-to-camera VLC systems that deploy a micro-screen. [14] adopts an active matrix LED (AMLED) micro-screen (12mm x 7.2mm, or 2.87 inches) as the transmitter, but they do not use a camera as a receiver. Instead, an advanced photo-receiver (New Focus 1801) is used which achieves a data rate of 1.25 Mbps over a distance of 25 cm. The New Focus 1801 photo-receiver cost over a thousand Euros (€1392) while our work uses off-the-shelf smartphones. Note that the AMLED micro-screen in [14] (2.87 inches, 1.17 W) is considerably larger in size and consumes more energy, compared to the FLCoS micro-screen in this work (0.2 inches, 180 mW).

Discussion

The screen-to-camera VLC systems discussed above are the most related works. However, most of them use a large-size screen like a computer monitor, while our work deploys a micro-screen. A large screen allows encoding much more data in a single frame for the same code density. And for the same amount of information, larger screens are easier for a camera to capture since it emits more light, and hence results in a longer operating distance. On the flip side, a monitor consumes much more power, between 20 to 100 W. The micro-display, in our case, only consumes 180 mW.

Some of the studies utilize a professional DSLR camera or an advanced photo-receiver, which are strongly capable of capturing light, either on sensitivity or speed. This superiority directly reflects on the high success rate of capturing high-capacity 2D barcodes and the high frame rate it maintains. With this project, cameras equipped in off-the-shelf smartphones are utilized.

This work is one of the very few that exploits micro-screen-to-camera links for VLC. The above studies inspire this project in many ways, but none of them shares the same conditions with us. Our goal is to achieve a reliable

	Screen size	Receiver model	Range	Data rate
Pixnet [18]	30-inch monitor	Casio EX-F1 (DSLR camera)	10 m	12 Mbps
Focus [7]	Monitor (exact size unknown)	Samsung Galaxy S6 Google Nexus One	35 cm	240 kbps
Chromacode [31]	27-inch monitor	Google Nexus 6P	50 cm	700 kbps
Cobra [6]	4-inch smartphone screen (Google Nexus S)	HTC Inspire	12.7 cm	172 kbps
RDCode [24]	7-inch tablet screen (Google Nexus 7)	Google Nexus 4, Samsung Galaxy S3	21 cm	174.4 kbps
RainBar [25]	Unknown	Smartphone (model unknown)	12 cm	955.68 kbps
[14]	2.87-inch AMLED micro-display (12 mm * 7.2 mm)	New Focus 1801 (Photo-receiver)	25 cm	1.25 Mbps
This work	0.2-inch FLCoS micro-display	Redmi5A Pixel2XL	12 cm	11.4 kbps

Table 2.1: **Summary of the most related screen-based VLC systems.** The data rate of these related works are higher than our work. One major reason is that most of them use screens in significantly large sizes. The other reason is that some of the works use a high-end DSLR camera [18] or an advanced photo-receiver [14].

and high-speed communication link between a FLCoS micro-display and off-the-shelf smartphones. Besides, we want to build a compact-size, self-contained embedded system with a practical application.

Chapter 3

Improving the data rate

This chapter explains how this work improves the data rate by reducing the transmission and processing times. *First*, the basic transmission sequence is described in Section 3.1, along with the message composition and data rate calculation. *Second*, Section 3.2 explains how the implementation of real-time image reading and processing decreases the reception time. *Last*, Section 3.3 depicts how the transmission time is shortened by increasing the transmission rate and deploying 2 transmitters.

3.1 The transmission sequence

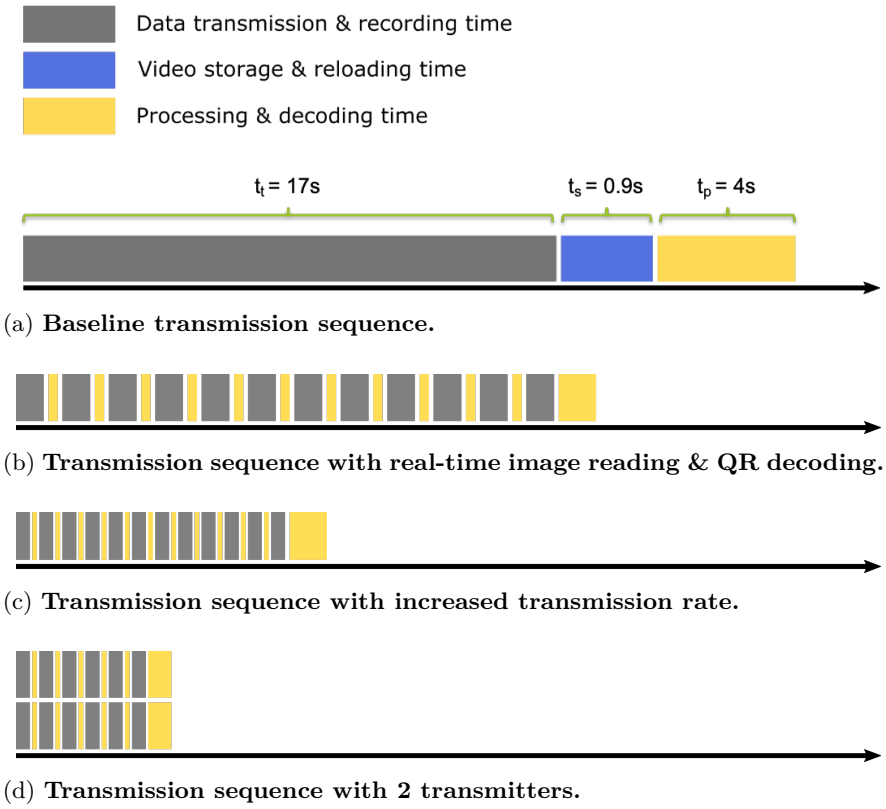
A basic data transmission process is visualized in Figure 3.1a, where the transmitter displays a sequence of QR codes in loops while the receiver (smartphone) records a video for the same duration as the source video. After the recorded video is saved and reloaded, the smartphone takes some time to process the video frames and decode the message.

The composition of the transmitted message and the calculation of data rates are explained in the Section 3.1.1 and 3.1.2.

3.1.1 Message composition

QR codes are used as packets in data transmissions because of their high-density capabilities. It also contains error correction and is resilient to image distortion caused by the camera angle and low contrast. The QR code image generator in Python [20] is used to generate the QR code sequence. Currently, the message contains 17-byte QR codes (version 1) where the first byte is the sequence number.

In early tests, the success rate of decoding a sequence of QR codes was found to be about 80% (Figure 3.2a). To ensure a successful data transmission process, some QR codes containing Reed-Solomon codes are added to the message (Figure 3.2). Although the addition of Reed-Solomon codes introduces redundancy to the data, it increases the success rate of message decoding. This project uses a Java implementation of Reed-Solomon code from Backblaze [11]. The whole message is configured to have 255 QR codes, where 75% (191) are QR codes with actual data, and the remaining 25% (64) contain Reed-Solomon codes. In



(a) **Baseline transmission sequence.**

(b) **Transmission sequence with real-time image reading & QR decoding.**

(c) **Transmission sequence with increased transmission rate.**

(d) **Transmission sequence with 2 transmitters.**

Figure 3.1: **Summary of transmission sequences.** For the baseline transmission in (a), the message consists of 255 QR codes transmitting at 15 FPS, which takes $t_t = 17$ seconds for transmission. On average, the smartphone (*Redmi5A*) recording a video of the message takes 0.9 seconds for storage (t_s) and 4 seconds for processing (t_p).

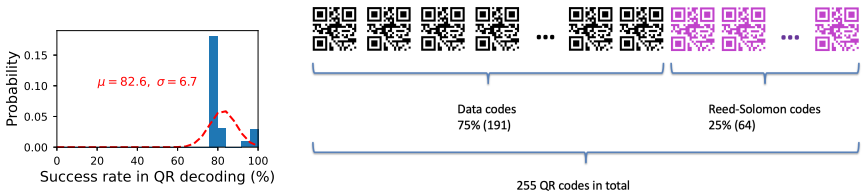
this case, the message is recoverable if 75% (191) QR codes are detected during transmission.

The first byte of each QR code contains the sequence number, hence, the locations of undetected packets in the message are known and marked as *erasures* (errors with known location). And the amount of erasures that the system can tolerate is equal to the amount of redundancy ($64 * 17 = 1088$ bytes).

3.1.2 Data rate calculation

Throughput and goodput are the two metrics of interest for quantifying data rate improvement. This VLC system is a simplex communication channel and packets may be lost during transmission.

- The **throughput** is the total amount of data in the detected QR codes divided by the transmission duration.
- For successful transmissions, the **goodput** is the amount of application-



(a) **QR success rate.** (b) **Configuration of data codes and Reed-Solomon codes.**

Figure 3.2: **Message composition.** Base on the QR success rate in (a), the transmitted message is configured to have 255 QR codes where 75% are data codes and 25% are Reed-Solomon codes.

level data divided by the transmission duration.

The calculation of throughput and goodput can be formulated as (3.1) and (3.2). Notice that the first byte of every QR code is reserved for the sequence number and it is excluded when calculating the goodput.

$$Throughput = \frac{Q * B * 8}{t_t + t_s + t_p} \ [bps], \tag{3.1}$$

$$Goodput = \frac{K * (B - 1) * 8}{t_t + t_s + t_p} \ [bps], \tag{3.2}$$

where:

- Q: the number of decoded QR codes in one transmission,
- B: the number of bytes in one QR code,
- K: the number of QR codes that stores actual data,
- t_t, t_s, t_p: the time duration (in seconds) for transmission, storage and processing respectively.

3.2 Reducing the reception time

From the basic transmission sequence in Figure 3.1a, two issues on the receiver side can be found:

1. *A noticeable amount of time is spent on saving and reloading the recorded video file.*
   For example, it takes 0.9 seconds for the `Redmi5A` to save and reload a 17-second video.

2. *Processing of video frames is only possible after the complete video is recorded and reloaded.*

Observation

The first issue is addressed by replacing the video recording with the implementation of real-time image reading. With real-time image reading, images frames captured by the camera are stored in memory instead of saving to device storage. Hence the time for saving and reloading a video file is removed ($t_s = 0$).

The second issue is solved by adopting a priority-based multi-threading scheme that enables real-time image processing and QR decoding. To capture image frames continuously at high speed, the Android phone is programmed to have a camera exposure time shorter than the duration for one frame. Thus there are idle time slots between capturing each frame. The multi-threading scheme allows the smartphones with multi-core CPUs (4 cores for `Redmi5A` and 8 cores for `Pixel2xl`) to utilize the camera idle times. The transmission sequence is now similar to Figure 3.1b, the processing is done in between camera idle times and the total data reception time is reduced noticeably.

The implementation details are presented in Section 3.2.1 and 3.2.2.

3.2.1 Real-time image reading

To remove video recording, this project utilizes the `ImageReader` class in Android [10]. The `ImageReader` class allows direct application access to the image data in real time, hence accelerating the data transmission process. This project uses the OpenCV library for real-time image processing. Images received from `ImageReader` need to be converted to OpenCV `Mat` format. To reduce the processing burden and ensure a fluent transmission flow, the Android app detects a region of interest (ROI, a region in the image that only contains the transmitter) at start-up.

Convert YUV image to OpenCV Mat

Everytime when an image is available, the callback function `onImageAvailable()` will be invoked. The image data is in format `YUV_420_888`, one of the most common image formats supported by Android cameras [17]. This type of image has one plane for luminance (Y) and two chrominance plane: blue projection (U), red projection (V) [27]. The `YUV_420_888` image need to be converted to an OpenCV `Mat` for further processing. The data transmission process only considers the QR code pattern, but not the color it contains. To reduce the burden on the phone, only the luminance data (Y plane) is saved to construct a gray-scale OpenCV `Mat`. An example Java code for such a conversion is shown below:

```
1  private static ArrayBlockingQueue<Mat> images = new
       ArrayBlockingQueue<Mat>(CAPACITY);
2
3  public void onImageAvailable(ImageReader imageReader) {
4      // Acquire image.
5      Image img = imageReader.acquireNextImage();
6
7      // Fetch y channel data only.
8      ByteBuffer yPlane = img.getPlanes()[0].getBuffer();
9      int ySize = yPlane.remaining();
10     byte[] byteData = new byte[ySize];
11     yPlane.get(byteData, 0, ySize);
12
```

```
13      // Convert image data to a OpenCV Mat object.
14      int height = img.getHeight();
15      int width = img.getWidth();
16      Mat mY = new Mat(height, width, CvType.CV_8UC1);
17      mY.put(0, 0, byteData);
18
19      images.add(mY);
20  }
```

ROI detection

At the beginning of each transmission, the Android application detects a region of interest (ROI) from the received images (as `Rect`). The ROI is an area that contains only the QR code area in the captured image. This is achieved by 1) applying binarization to the image, 2) then finding contours in the binarized image, and finally, after detecting the contours, 3) selecting and returning the region that:

- has 4 corners (quadrilateral),
- has an area whose percentage over the image lies in a predefined range ($[1/50, 1/3]$),
- has an aspect ratio that is within a predefined range ($[0.5, 2]$).

This detection is formulated in Algorithm 1.

Algorithm 1 Detect a region of interest

```
procedure DETECTROI(mat, roi)
    binMat ← Binarize mat;
    contours ← find contours in binMat;
    for all contour ∈ contours do
        if contour has four corners then
            if minimum area < area of contour < maximum area then
                if minimum aspect ratio < aspect ratio of contour < maximum aspect ratio then
                    roi ← contour;
                    break;
                end if
            end if
        end if
    end for
end procedure
```

3.2.2 Real-time QR decoding

A priority-based multi-threading scheme

While the `ImageReader` is continuously reading and saving images, multi-threading is exploited to enable further processing of those saved images and decoding the QR in the meantime. This is achieved with the use of the Java class

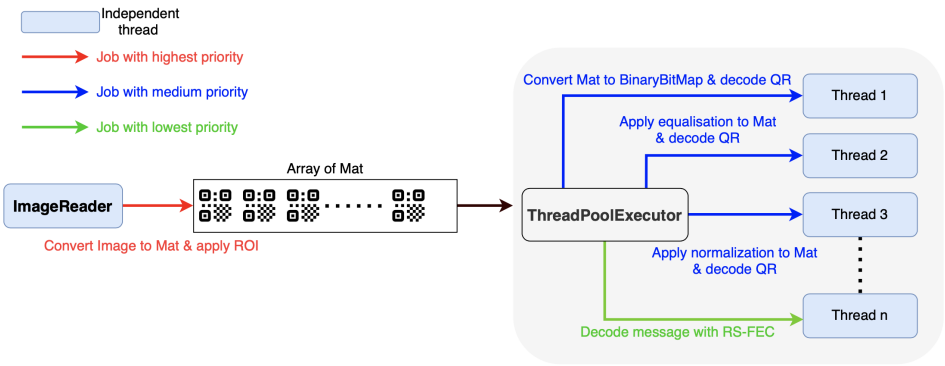


Figure 3.3: **The multi-threading scheme.** Image data available from the `ImageReader` are converted to OpenCV `Mat` format for further processing. Each `Mat` image is applied with 3 different image processing techniques to improve the reliability of QR decoding.

`ThreadPoolExecutor` [23]. A unit of operation that performs on the smartphone is called a *job*. The `ThreadPoolExecutor` manages the execution of a queue of jobs within a fixed number of threads. In this case, the number of threads is set to the number of processing cores of the running smartphone. To ensure proper execution while making full use of the processing cores on the phone, different jobs are assigned with different priorities. As in Figure 3.3, there are three types of jobs in the Android application:

- **Highest priority job** (red arrow): for every captured image (`YUV_420_888`), convert it to `Mat`, apply ROI and store in memory,
- **Medium priority job** (blue arrow): for every captured image, apply processing (*equalization* and *normalization*) and decode QR,
- **Lowest priority job** (green arrow): when enough images are captured, decode the whole message with Reed-Solomon forward error correction.

To improve the success rate of decoding a QR, *normalization* and *equalization* will be applied to each `Mat` image. Thus for each QR image, there will be three versions that are sent to a QR decoder:

1. the grayscale version
2. the equalized version
3. the normalized version

The QR code decoder in use is from **ZXing** [32]. The `Mat` image needs to be converted to a `BinaryBitmap` before inputting it to the ZXing QR decoder. The ZXing QR decoder will return a `Result` if it detects the QR code.

Stop receiving images when a minimum number of QR codes are detected

According to the **Nyquist Sampling Theorem**, the camera should capture images at a frequency that is twice as high as the transmitting frequency. To

receive all the transmitted information, the camera capture session should last for the same duration as the transmitted message. For example, when the transmitter sends 255 QR codes at 15 FPS (which takes 17 seconds), the smartphone captures images at 30 FPS for 17 seconds. However, as described in Section 3.1.1, the transmitted message contains 255 QR codes, where 75% (191) are QR codes with actual data, and the remaining 25% (64) contain Reed-Solomon codes. Detecting 75% (191) of the unique QR codes, regardless of their contents, means the complete message is recoverable. Thus, while the `ImageReader` is receiving images, the Android application is detecting QR code from each saved image and updating a counter. Once sufficient (191) QR codes are detected, the program will stop the `ImageReader` from receiving images, which decrease both the reception time (t_t) and processing time (t_p) by 25% on average.

3.3 Reducing the transmission time

To further reduce the time on sending messages, there are two things that can be improved on the transmitter side:

1. *'Violating' Nyquist Sampling Theorem and increase the transmission rate.* Up to this point, the camera devices are receiving information at 30 FPS, Based on the Nyquist Sampling Theorem, the transmission rate is set to half of the receiving rate (15 FPS). The transmission rate can be increased to reduce the delay, and packet loss is not a major problem because the integrated Reed-Solomon code will allow us to recover them. Hence there is no need to limit the transmission rate to half of the reception rate.

2. *Widen channel bandwidth by deploying 2 transmitters.* Currently, only one transmitter is deployed. As shown in Figure 3.5, a significant portion of the camera preview is idle. Adding another transmitter to the system will make better use of the camera sensor and broaden the channel bandwidth.

Observation

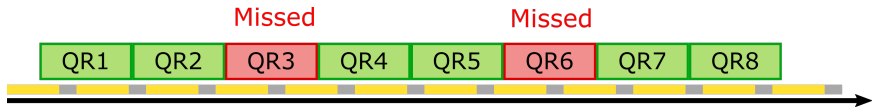
The transmission sequences after the above two adjustments are visualized in Figure 3.1c and 3.1d. By increasing the transmission rate from 15 FPS to 30 FPS, the data transmission time (t_t) decreased by 50%. We then further increased both the transmission and reception rate to 60 FPS (the highest achievable frame rate of the system). More detailed descriptions are contained in Section 5.2. Additionally, the same amount of data are evenly distributed to the two transmitters, which results in a transmission time (t_t) that is half of the previous. Implementation details are presented in Section 3.3.1 and 3.3.2.

3.3.1 Increase transmission rate

As shown in Figure 3.4, if the reception rate is set to twice the transmission rate. Every QR code is captured since at least one camera frame is exposed only to that QR code. In most cases, a QR code is captured twice. If the reception is equal to the transmission rate, some QR codes are missed because no frame is exposed only to them. This is not a major problem since the integrated Reed-Solomon code allows the decoder to recover those missed QR codes. Hence,

starting from a 15-FPS transmitter and a 30-FPS receiver, we first increase the transmission rate from 15 to 60 FPS with a step size of 5 FPS. This adjustment showed that the system work the best with the same transmission and reception rate (both at 30 FPS). Based on this, we further increase both transmission and reception rate to 60 FPS (the highest frame rate the system can obtain) to achieve the fastest transmission time (t_t = 2.2 seconds). Detailed evaluation results are presented in Section 5.2.

(a) **Reception rate equals twice of the transmission rate.**



(b) **Reception rate equals the transmission rate.**

Figure 3.4: **Increasing the transmission rate.** In (a), every QR code is captured since at least one of the frames is exposed only to that QR code. In (b), some QR codes are missed if no frame is exposed only to them. This is not a problem since the integrated Reed-Solomon code allows the decoder to recover those missed QR codes.

3.3.2 Deployment of 2 transmitters

To utilize the camera sensor more efficiently, another transmitter (FLCoS) is added to the system. The adjustments required for both the transmitter side and receiver side are elaborated on below.

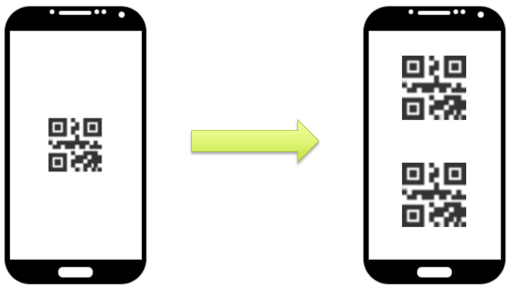


Figure 3.5: **Widen channel bandwidth by deploying two transmitters.** To better utilize the idle portion in camera preview, another transmitter is added and the channel bandwidth is doubled.

Divide QR sequences into half

The same amount of information is used for transmission, which is evenly distributed to the two transmitters. This is implemented by slicing the source video file into two, each fed to one transmitter. An example Python code to generate two video files from text messages is included in Appendix B.

2-ROI detection in Android application

To ensure that the image processing techniques (*equalization* and *normalization*) contribute to the QR decoding process, the detection of 2 ROIs is required. Similar to detecting a single ROI in Section 3.2.1, the Android application detects two ROIs by first applying binarization of the image, then selecting two contours that contain the transmitters. The difference, in this case, is to find two contours in the same image. The detection of 2 ROIs is formulated in Algorithm 2.

Algorithm 2 Detect 2 regions of interest

```
1:  procedure DETECTROI(mat, rois)
2:      binMat ← Binarize mat;
3:      contours ← find contours in binMat;
4:      for all contour ∈ contours do
5:          if contour has four corners then
6:              if minimum area < area of contour < maximum area then
7:                  if minimum aspect ratio < aspect ratio of contour < maximum aspect ratio then
8:                      rois ← contour;
9:                      if length of rois == 2 then
10:                         break;
11:                     end if
12:                 end if
13:             end if
14:         end if
15:     end for
16: end procedure
```

Chapter 4

Self-contained embedded system

This section describes the solutions to the second research question, designing an independent system and demonstrating a use case. We aim to build a tiny box that can be deployed anywhere without the need for external power or data cables. The system provides useful information for users depending on the scenarios. The prototype design of a stand-alone embedded system is presented in Section 4.1. In Section 4.2, an Android application integrated with a map view demonstrates a potential use case.

4.1 Hardware setup

To achieve an independent system that operates on its own, two limitations need to be tackled:

1. *Remove the need of a laptop as the external source of data.* Currently, a laptop is required to provide video signals. Besides, the micro-display module only accepts composite video signals in *National Television System Committee* (NTSC) standard, which is not supported by the laptop. Thus extra conversions from the HDMI interface to NTSC standard.

2. *Remove the need of an external power source.* Apart from data sources, the transmitters also reply on the external laptop for power supplies.

The solution is to add embedded processing units (Raspberry Pi Zero), a solar cell and a battery to the system. The following subsections introduce the system design and describe the selected hardware components.

4.1.1 System design

A diagram of the hardware system is shown in Figure 4.1. This design consists of a solar harvesting panel, a lithium battery, two Raspberry Pi Zero W and 2 sets of FLCoSs (with controller modules). The solar panel can be used to charge the battery when exposed to sunlight. The battery is able to output 5 volts to both Raspberry Pi development boards. When powered on, each Raspberry Pi

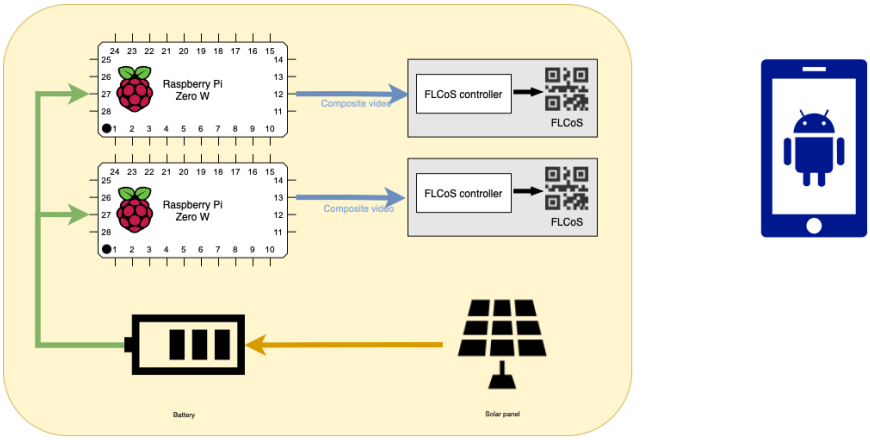


Figure 4.1: **Hardware system design.**

Zero W can directly transmit composite video signal (NTSC) to the connected FLCoS display modules. To interact with such a system, just turn on the power switch and use an Android phone (with the dedicated Android application) to scan the transmitters. The received data will be displayed on the smartphone.

4.1.2 Hardware components

The power specifications and prices of the selected hardware components are summarized in Table 4.1.

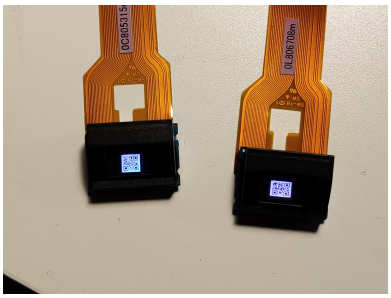
	Power generation	**Power consumption**	**Price**
FLCoS micro-display module	-	180 mW	$70
Raspberry Pi Zero W	-	0.4 W	€11.5
Battery and DC/DC converter module	~ 5 W	~ 5 W	$14.95 + $19.95
Solar panel	Max 5.71 W	-	€79.07

Table 4.1: **Power specifications and prices of the hardware components.**

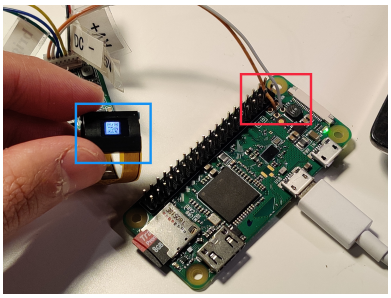
Here are more detailed descriptions of the components:

1. *Micro-display module.*
   This is a low power ferroelectric liquid crystal (FLC) reflective micro-display panel from Control Electronics Co., Ltd, with a nominal power consumption of 180 mW. It fits this project for also its tiny size (0.2 inches diagonal) and high resolution (720*540). It is commonly used for helmet-mounted displays and video glasses.

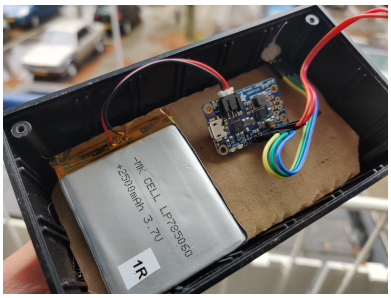
2. *Raspberry Pi Zero W.*
   This is a small-size (66 mm x 30.5 mm x 5 mm) single-board computer with a 1 GHz single-core CPU and 512 MB memory. The key points are that it can output composite video signal (NTSC) directly to the micro-display module while being power efficient (as low as 80 mA or 0.4 Watt). It also has an affordable price (11.5 euros) for small size embedded systems.



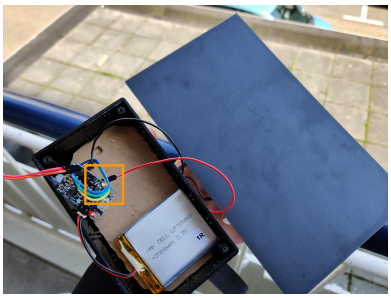
(a) **Micro-display.**



(b) **Raspberry Pi Zero W.**



(c) **Battery module.**



(d) **Solar panel.**

Figure 4.2: **Hardware components.**

3. *Battery and DC/DC boost converter (Adafruit Powerboost 1000C).*
   A lithium-ion polymer battery (LP785060) is chosen for this project because of its high capacity (2500 mAh) and lightweight (43 g). This battery outputs 3.7 V in nominal while the Raspberry Pi Zero W requires 5 V input. Thus a DC/DC boost converter is needed. The battery module (including the DC/DC boost converter) can output at 5 W (5 V, 1 A). The battery module is also required for charging the battery using the solar panel.

4. *Solar panel.*
   The selected solar panel is an IXYS SMD262K10L consisting of 10 solar cells with a cell efficiency of 25%. The power generated by this solar panel is 5.71 W (5.58 V, 1.02 A) at maximum, which fulfils the requirements of 4.75-5.25 V for charging the battery. The size of this 10-cell panel is comparable to the other components (220 mm x 126 mm x 2.1 mm). A smaller panel with fewer solar cells will not be able to generate enough power to charge the battery.

4.2 The demonstration application

A potential use case of this VLC system is shown in Figure 4.3, where there is no internet connection. Consider a tourist looking for information about restaurants, museums or other events. The user does not rely on other wireless

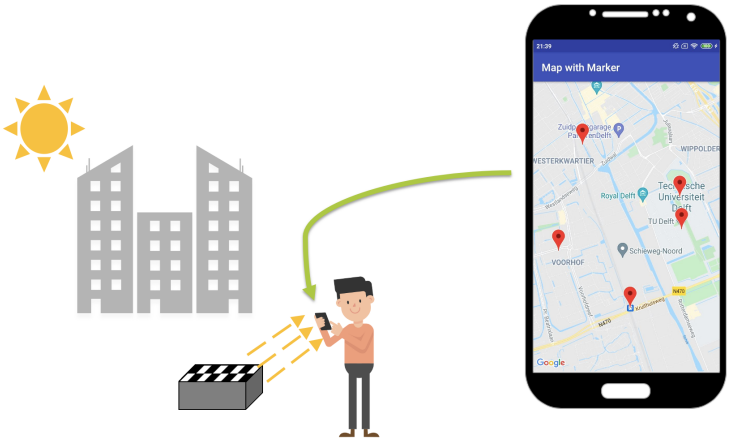


Figure 4.3: **Transmit locations and show them on a map.**

connections and simply puts the phone on top of the box, the information is transmitted.

The Android application **DMDFLCQR** presented with this project is available on Github. Here are the instructions on how to use a smartphone installed with the Android application to scan and decode information from the presented micro-screen-to-camera system (screenshots of each instruction are listed in Figure 4.4):

(a) *Open the Android application.*
The app shows a preview of the camera. Notice that the `SCAN` button is hidden if the transmitters are not detected.

(b) *Point the camera to the transmitters and preferably keep the distance between and smartphone and transmitters around 12 cm.*
Once the transmitters are detected, the ROIs will be shown in the camera preview and the `SCAN` button becomes visible.

(c) *Keep the transmitters within the ROIs and press the `SCAN` button.*
After scanning and decoding, the received locations will be shown as markers in a map view.

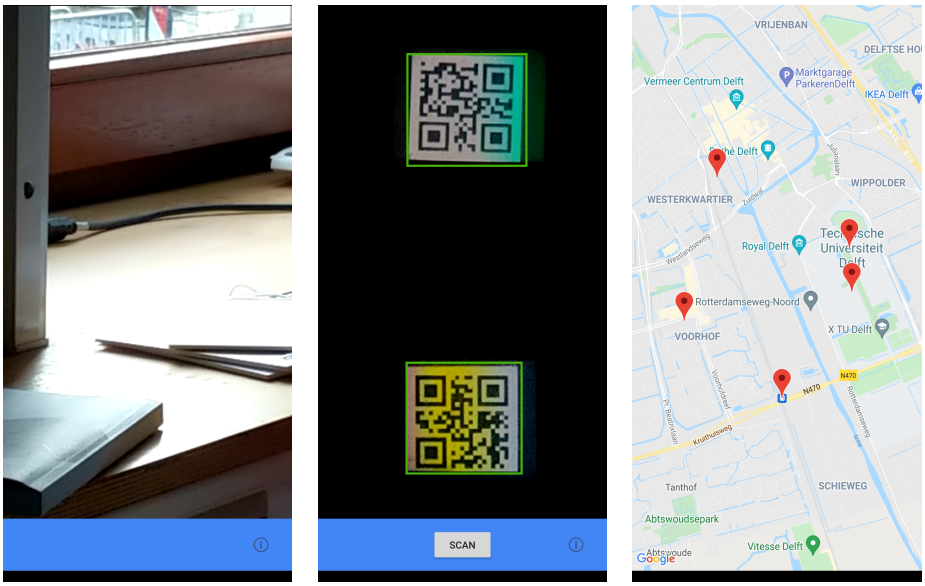
To build this application, two adjustments are needed:

1. *Encode coordinates into the transmission message*,
2. *Integrate GoogleMap to the Android application.*

The implementation details are presented in Section 4.2.1 and 4.2.2.

4.2.1 Message with coordinates

Up to this point, this system is only transmitting text messages. The text messages are encoded into QR codes in binary mode. For the demonstration application, geographical coordinates (type `LatLng` in Java) are also encoded in binary mode. A `LatLng` consists of a longitude (8-byte `double`) and a latitude



(a) (b) (c)

Figure 4.4: **Screenshots of the Android application.**

(8-byte `double`). The receiver application is aware that one coordinate occupies 16 bytes of data (equivalent to the capacity of the QR code version 1 excluding a 1-byte sequence number) and decode the coordinates accordingly.

4.2.2 Show markers in the map

Coordinates received by the smartphone are represented as markers in a map view. The GoogleMap Software Development Kit (SDK) is added to the Android application for showing a map view. An example code to show markers in a map view is:

```
public void onMapReady(GoogleMap googleMap) {
    // Construct markers
    ArrayList<LatLng> markers = new ArrayList<>();
    int i = 0;
    while (i < 10) {
        markers.add(new LatLng(
                // Longitude
                Double.parseDouble(mParam_coordinates.get(i++)),
                // Latitude
                Double.parseDouble(mParam_coordinates.get(i++))
        ));
    }
    // Show markers
    showMarkers(googleMap, markers);
    }
}

private void showMarkers(GoogleMap googleMap, ArrayList<LatLng>
    markers) {
    double lat = 0;
    double lon = 0;
```

```
21      for(int i = 0; i < markers.toArray().length; i++) {
22          lat += markers.get(i).latitude;
23          lon += markers.get(i).longitude;
24          googleMap.addMarker(new MarkerOptions().position(markers.
        get(i)).title("Marker " + i));
25      }
26      LatLng center = new LatLng(
27              lat / markers.toArray().length,
28              lon / markers.toArray().length
29      );
30      googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(center,
        14));
31  }
```

Chapter 5

Evaluation

5.1 Experimental setup

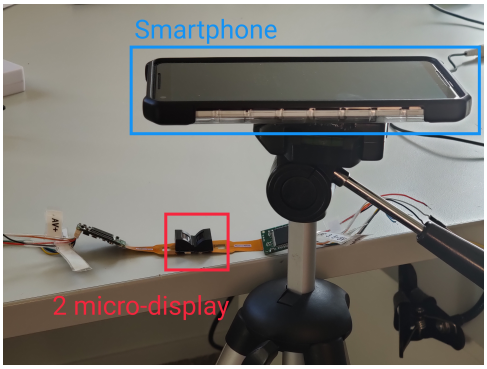


Figure 5.1: **Experimental setup.** The 2 FLCoS micro-display modules are placed next to each other on the table. The receiver smartphone is placed on a tripod, about 12 cm away from the transmitters.

The experimental setup is shown in Figure 5.1, where the two micro-displays are placed next to each other on the table. The smartphone is placed on a tripod, about 12 cm away from the micro-displays. We evaluated the system by 1) displaying information on the micro-display(s) continuously, 2) using the smartphones installed with the dedicated Android application to receive the message and calculate the data rates. The same amount of data are transmitted (as described in Section 3.1.1), only in different frame rates. The experiments are divided into five steps:

(i) ***1 TX (Baseline)***: One transmitter at 15 FPS and a *Redmi5A* smartphone recording a video at 30 FPS.

(ii) ***2 TX***: Two transmitters at 15 FPS and the *Redmi5A* recording a video at 30 FPS.

(iii) ***2 TX + Real-time***: Two transmitters at 15 FPS and the *Redmi5A* smartphone reading images in real time at 30 FPS.

(iv) ***2 TX + Real-time + 30 FPS***: Both transmission and reception rates are at 30 FPS, the *Pixel2xl* is used as receiver.

(v) ***2 TX + Real-time + 60 FPS***: Both transmission and reception rates are at 60 FPS, the *Pixel2xl* is used as receiver.

For each step, the transmission process is repeated 10 times. The individual results for each one of the five steps are described in Section 5.2.

5.2 Evaluation results

This section quantifies the data rate improvements step by step, which can be summarized in Table 5.1. For each step, the reduction in time obtained by applying the different steps is also summarized in Table 5.2. And more detailed explanations are presented in the following subsections.

	Throughput (kbps)	**Goodput (kbps)**
1 TX (Baseline)	1.5	1.1
2 TX	2.6 (*+73%*)	1.9 (*+73%*)
2 TX + Real-time	3.8 (*+46%*)	3.6 (*+89%*)
2 TX + Real-time + 30 FPS	7.5 (*+97%*)	7.0 (*+94%*)
2 TX + Real-time + 60 FPS	11.4 (*+52%*)	10.7 (*+53%*)

Table 5.1: **Summary of data rates.** The improvements in percentage are obtained by comparing with the prior step.

	Recording & saving time (s)	Processing time (s)	Total transmission time (s)	Detected QR codes
1 TX (Baseline)	17.9	4.0	21.9	247.0
2 TX	9.0	3.5	12.5	240.7

(a) **Summary of recording and processing time. These data are obtained with the Android application that requires recording a video.**

	Image reading & processing time (s)	Additional processing time (ms)	Total transmission time (s)	Detected QR codes
2 TX + Real-time	6.6	99.3	6.7	191.2
2 TX + Real-time + 30 FPS	3.3	69.4	3.4	192.5
2 TX + Real-time + 60 FPS	2.2	66.1	2.3	192.0

(b) **Summary of image reading and processing time. These data are obtained with the Android application that implements real-time image reading.**

Table 5.2: **Summary of transmission and processing time.**

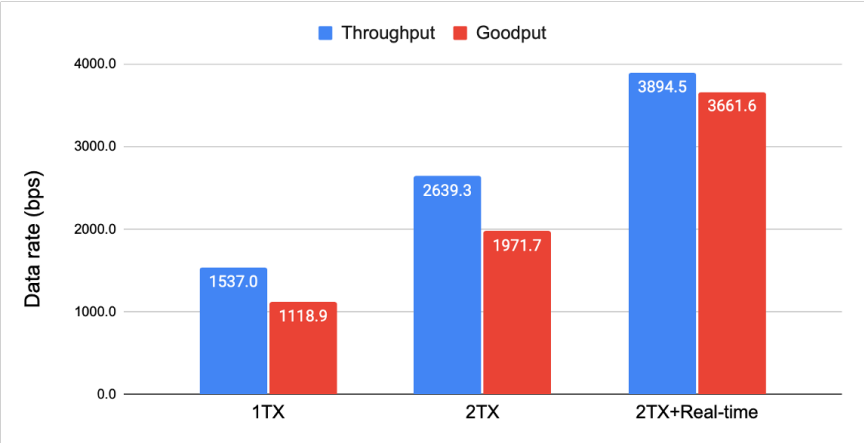


Figure 5.2: **Data rate improvements by deploying 2 transmitters and real-time image reading.** The First 2 columns indicate data rates with a 15-FPS transmitter and a receiver recording a video at 30 FPS. The 2 columns in the middle indicate data rates with two 15-FPS transmitters and a receiver recording a video at 30 FPS. The last two columns indicate data rates with two 15-FPS transmitters and a receiver reading images at 30 FPS in real time.

5.2.1 Baseline performance

According to the Nyquist Sampling Theorem described in Section 2.3.1, the receiver should operate at twice the frame rate of the transmitter. Thus the baseline experiment is set up as: the transmitter sends 255 QR codes at 15 FPS (which lasts 17 seconds) in loops and the receiver (Redmi5A) records a video at 30 FPS for 17 seconds. After that, the video is saved, reloaded and processed.

Results: Experiment results (Table 5.2) show that on average, 17.9 seconds were taken for the receiver to record and save the video and 4.0 seconds for processing. This means that almost a second (0.9) is spent on saving and reloading of the video file. On average, 247.0 QR codes are detected for each transmission.

The system can obtain **1.5 kbps** throughput and **1.1 kbps** goodput in this baseline performance.

5.2.2 Improvement by deploying two transmitters

The transmission message is split into two when deploying two transmitters. Each transmitter displayed half of the message at 15 FPS (which take $17/2 = 8.5$ seconds) in loops. This adjustment reduces the recording time to half and hence the receiver (Redmi5A) records a video at 30 FPS for 8.5 seconds.

Results: On average, 9.0 seconds were taken for the receiver to record and save the video and 3.5 seconds for processing. The time spent on saving and reloading a video file now becomes 0.5 seconds. On average, 240.7 QR codes are detected for each transmission, which means that not too many packets are lost compared to the baseline.

With 2 transmitters displaying data at the same time, the system is able to increase the throughput and goodput to **2.6 kbps** (73% improvement) and **1.9 kbps** (73% improvement) respectively.

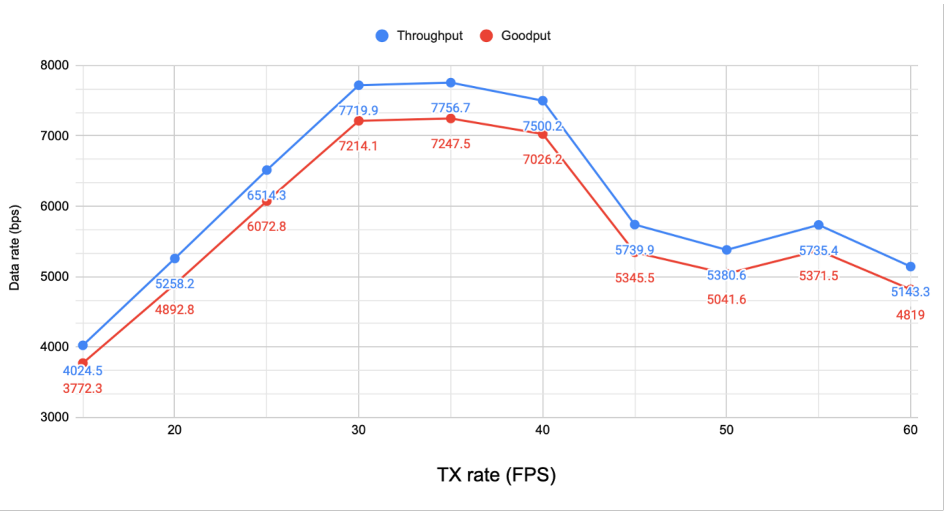


Figure 5.3: **The data rate changes when varying the transmitter's frame rate.** Keeping the reception rate at 30 FPS, vary the transmission rate from 15 to 60 FPS, the highest data rates are obtained when the transmission rate is equal to the reception rate (30 FPS).

5.2.3 Improvement by real-time image reading and parallel QR decoding

For a basic transmission sequence, the information is recorded as a video with the smartphone camera. After the video is saved and reloaded, the smartphone takes some time to process the video frames and decode the information.

- **Real-time image reading and processing.**
  We notice a certain amount of time is spent on saving and reloading the video, especially for resource-constrained smartphones (e.g., Redmi5A). This time depends on the size of the video file. For a 17-second recording, it takes 0.9 seconds for the Redmi5A to save the video to storage. For an 8.5-second recording, 0.5 seconds is consumed. To remove the requirement of recording a video, an `ImageReader` object in Android is utilized to enable real-time image reading. Image data are saved in memory for real-time processing.

  Since the Android application has real-time access to image data, a multi-threading scheme is adopted to process and decode QR codes once an image is available. The processing of images can start once the Android application starts running without the need to wait for video recording.

  Intermediate results: The time spent on saving and reloading the video file (t_s) is eliminated (0.5 seconds). The processing time is performed in between camera idles times.

- **Stop image reading when enough QR codes are detected.**
  Unlike video recording, real-time image reading allows keeping track of the total number of detected QR codes at run time. And thanks to the added Reed-Solomon codes, the whole message is recoverable when 75% (191)

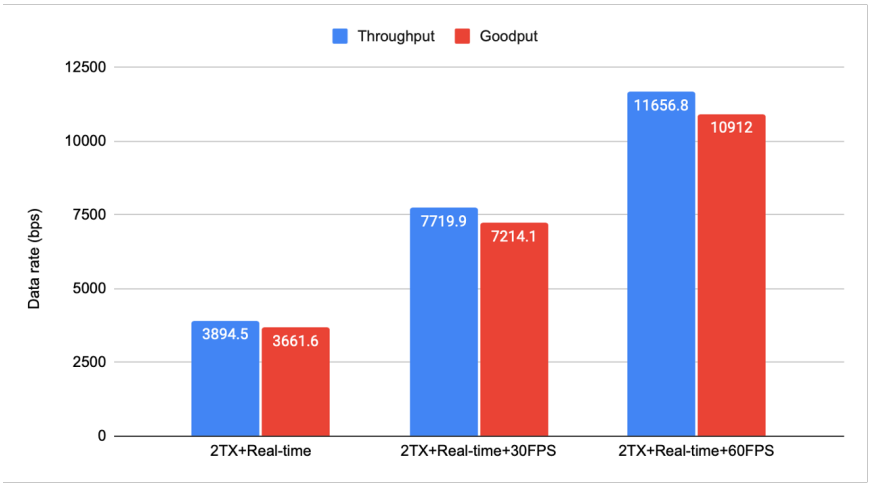


Figure 5.4: **Data rate improvement by increasing both transmission and reception rates.** The First 2 columns indicate data rates with two 15-FPS transmitters and a 30-FPS receiver. The second 2 columns indicate data rates with both transmission and reception rates at 30 FPS. The last 2 columns indicate data rates with both transmission and reception rates at 60 FPS.

QR codes are detected. To further reduce the data transmission time, a real-time checking mechanism is implemented in the Android application such that it stops reading images when enough QR codes are detected.

Intermediate results: The experiment results (Table 5.2) confirm that on average 191.2 QR codes are detected in every transmission. Compared to the Android application that requires recording the entire video (Section 5.2.2), the time duration for detecting $240.7 - 191.2 = 49.5$ QR codes is saved. This is at least 3.3 seconds considering lost packets and undetectable QR codes. Together with the above adjustment, the transmission time (t_t) is further reduced by 27% (from 9 to 6.6 seconds).

Results: By implementing the real-time image reading and processing, the average total transmission time (t_t,) is reduced from 12.5 to 6.7 seconds (Table 5.2). The system is able to increase the throughput and goodput to **3.8 kbps** (46% improvement) and **3.6 kbps** (89% improvement) respectively, comparing to the prior step.

5.2.4 Improvement by increasing the transmission rate

The transmission and reception rates in the previous steps are all 15 FPS and 30 FPS respectively. In this subsection, we exploited faster transmission by **first**, varying the transmission rate from 15 to 60 FPS and **second**, increase both transmission and reception rates to 60 FPS.

1. **Effect of increasing the transmission rate from 15 to 60 FPS.** As depicted in Figure 5.3, both the throughput and goodput have almost linear increments when the transmission rate is increased from 15 FPS

to 30 FPS. Although the date rates fluctuate a bit, it shows a downward trend when the transmission rate further rises from 30 FPS to 60 FPS.

Results: The best result happens when both the transmission rate and reception rate are 30 FPS. Experiment results (Table 5.2) show that when the transmission rate is doubled, the total transmission time is nearly halved (from 6.7 to 3.4 seconds). This means that, for our system, keeping the reception rate twice as high as the transmission rate is not efficient. The throughput and goodput reach up to **7.5 kbps** (97% improvement compared to prior step) and **7.0 kbps** (94% improvement compared to prior step) respectively.

2. **Effect of Increasing both the transmission and reception rate to 60 FPS.**
   Knowing that the best performance is obtained when the transmission rate and the reception rate are equal, we increase them both to 60 FPS, which is the highest frame rate the system (both transmitter and receiver) can achieve. For the transmitter, the frame rate of the micro-display is limited by the connected HDMI cable. For the receiver, the image reading rate is restricted by the camera's capability to maintain a proper exposure time.

   Results: By increasing both the transmission and reception rates to 60 FPS, the total transmission time was reduced by 48%, from 3.4 to 2.3 seconds. The throughput and goodput of this system eventually reach up to **11.4 kbps** (52% improvement compared to prior step) and **10.7 kbps** (53% improvement compared to prior step) respectively.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this work, we presented a standalone micro-screen-to-camera VLC system that can transmit information at a data rate as high as 11.4 kbps.

To improve the data rate, we *first* adopt several methods including a real-time implementation of image reading and packet decoding. *Secondly*, another transmitter is added to the system to increase the data rate.

To make the system self-reliant, we utilize two Raspberry Pi Zeros for feeding data to the micro-display. A high-capacity battery and a solar panel are selected as power sources. We also developed a demonstration application in Android that is able to receive coordinate information from the transmitter and show them as markers in a map view.

Although there are many works in the field of screen-to-camera communication, this project is one of the very few that explores micro-displays for near-distance communication. The adoption of a FLCoS micro-display implies the possibility of a passive VLC system. That is, the embedded LED can be replaced with ambient light or sunlight to reduce power consumption. We believe this work will motivate more researchers to explore visible light communication, especially on a passive micro-screen-to-camera link.

6.2 Future works

Although this work has improved the data rate of a micro-screen-to-camera system by over 7-fold, there is still a lot of potential for this communication link. Some possible future works are presented in the following sections.

6.2.1 Explore color shift keying with embedded LED

Cobra [6], *RainBar* [25] and *Styrofoam* [16] have proven the feasibility of transmitting and decoding 2D barcodes with colors from a smartphone screen to a camera. In this work, the transmitters are sending binary (black and white) QR codes, a promising idea would be sending color QR codes to exploit color-shift

keying. An example implementation is to generate QR codes in red (R), green (G) and blue (B). Theoretically, the data rate would be tripled since the data capacity of an RGB QR code is three times the binary one.

6.2.2 Custom 2D barcode

Many screen-to-camera VLC systems are paired with specially-designed 2D barcodes for data transmission. *Pixnet* [18], *Strata* [9] and *Focus* [7] utilize an gray-scale OFDM-based visual code that encodes data in the frequency domain for monitor-size screens. *Cobra* [6], *RainBar* [25] and *Styrofoam* [16] provide 2D barcodes with colors for smartphone screens. The packets for this system are QR codes with fixed density. Besides, QR codes are square which cannot fill up the entire micro-screen with a 4:3 aspect ratio. Hence using other types of 2D barcodes or designing a dedicated code pattern are worthy choices to exploit the channel bandwidth.

6.2.3 Duplex communication

The current system can only send information from the micro-screen to a smartphone. The communication system would be more interactive and well-rounded if it can send information in both directions. *RetroVLC* [13] presents a duplex system that pairs each transmitter with a photodiode to receive light information. A more relevant example [5] establishes a bi-directional VLC system between an LED and a smartphone. Apart from the LED-to-camera link, feedback signals can also be sent from the flashlight on the phone to the LED. Extending this work to be a duplex system would allow the user to manually select the desired information in real-time.

Bibliography

[1] *About LCD Displays*. https://www.yaoyulcd.com/whatislcd.htmll.

[2] Karan Ahuja et al. 'LightAnchors: Appropriating Point Lights for Spatially-Anchored Augmented Reality Interfaces'. In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST '19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 189–196. ISBN: 9781450368162. DOI: 10.1145/3332165.3347884. URL: https://doi.org/10.1145/3332165.3347884.

[3] Rens Bloom, Marco Zúñiga Zamalloa and Chaitra Pai. 'LuxLink: Creating a Wireless Link from Ambient Light'. In: *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. SenSys '19. New York, New York: Association for Computing Machinery, 2019, pp. 166–178. ISBN: 9781450369503. DOI: 10.1145/3356250.3360021. URL: https://doi.org/10.1145/3356250.3360021.

[4] Willy Anugrah Cahyadi et al. 'Mobile Phone Camera-Based Indoor Visible Light Communications With Rotation Compensation'. In: *IEEE Photonics Journal* 8.2 (2016), pp. 1–8. DOI: 10.1109/JPHOT.2016.2545643.

[5] Alexis Duquel et al. 'Decoding methods in LED-to-smartphone bidirectional communication for the IoT'. In: *2018 Global LIFI Congress (GLC)*. 2018, pp. 1–6. DOI: 10.23919/GLC.2018.8319118.

[6] Tian Hao, Ruogu Zhou and Guoliang Xing. 'COBRA: Color Barcode Streaming for Smartphone Systems'. In: MobiSys '12. Low Wood Bay, Lake District, UK: Association for Computing Machinery, 2012, pp. 85–98. ISBN: 9781450313018. DOI: 10.1145/2307636.2307645. URL: https://doi.org/10.1145/2307636.2307645.

[7] Frederik Hermans et al. 'FOCUS: Robust Visual Codes for Everyone'. In: MobiSys '16. Singapore, Singapore: Association for Computing Machinery, 2016, pp. 319–332. ISBN: 9781450342698. DOI: 10.1145/2906388.2906399. URL: https://doi.org/10.1145/2906388.2906399.

[8] Pengfei Hu et al. 'High Speed LED-to-Camera Communication using Color Shift Keying with Flicker Mitigation'. In: *IEEE Transactions on Mobile Computing* 19.7 (2020), pp. 1603–1617. DOI: 10.1109/TMC.2019.2913832.

[9] Wenjun Hu et al. 'Strata: Layered Coding for Scalable Visual Communication'. In: MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, pp. 79–90. ISBN: 9781450327831. DOI: 10.1145/2639108.2639132. URL: https://doi.org/10.1145/2639108.2639132.

[10] *ImageReader*. https://developer.android.com/reference/android/media/ImageReader. 2021.

[11] *JavaReedSolomon*. https://github.com/Backblaze/JavaReedSolomon. 2021.

[12] Ye-Sheng Kuo et al. 'Luxapose: Indoor Positioning with Mobile Phones and Visible Light'. In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, pp. 447–458. ISBN: 9781450327831. DOI: 10.1145/2639108.2639109. URL: https://doi.org/10.1145/2639108.2639109.

[13] Jiangtao Li et al. 'Retro-VLC: Enabling Battery-Free Duplex Visible Light Communication for Mobile and IoT Applications'. In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. HotMobile '15. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2015, pp. 21–26. ISBN: 9781450333917. DOI: 10.1145/2699343.2699354. URL: https://doi.org/10.1145/2699343.2699354.

[14] Xianbo Li et al. 'Design and Characterization of Active Matrix LED Microdisplays With Embedded Visible Light Communication Transmitter'. In: *Journal of Lightwave Technology* 34.14 (2016), pp. 3449–3457. DOI: 10.1109/JLT.2016.2562667.

[15] *LightView 201k Digital Display Module*. https://docplayer.net/165372082-Customer-product-specification-cps.html.

[16] Robert LiKamWa, David Ramirez and Jason Holloway. 'Styrofoam: A Tightly Packed Coding Scheme for Camera-Based Visible Light Communication'. In: VLCS '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, pp. 27–32. ISBN: 9781450330671. DOI: 10.1145/2643164.2643169. URL: https://doi.org/10.1145/2643164.2643169.

[17] Minhaz. *How to use YUV (YUV_420_888) Image in Android*. https://blog.minhazav.dev/how-to-convert-yuv-420-sp-android.media.Image-to-Bitmap-or-jpeg/. 2020.

[18] Samuel David Perli, Nabeel Ahmed and Dina Katabi. 'PixNet: Interference-Free Wireless Links Using LCD-Camera Pairs'. In: *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*. MobiCom '10. Chicago, Illinois, USA: Association for Computing Machinery, 2010, pp. 137–148. ISBN: 9781450301817. DOI: 10.1145/1859995.1860012. URL: https://doi.org/10.1145/1859995.1860012.

[19] *Polarization*. https://courses.lumenlearning.com/physics/chapter/27-8-polarization/.

[20] *Pure python QR Code generator*. https://github.com/lincolnloop/python-qrcode. 2021.

[21] Steven Ruzin. *Capturing images*. https://microscopy.berkeley.edu/courses/dib/sections/02Images/sampling.html. 2009.

[22] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. 5th. USA: Prentice Hall Press, 2010. ISBN: 0132126958.

[23] *ThreadPoolExecutor*. https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html. 2020.

[24] Anran Wang et al. 'Enhancing Reliability to Boost the Throughput over Screen-Camera Links'. In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, pp. 41–52. ISBN: 9781450327831. DOI: 10.1145/2639108.2639135. URL: https://doi.org/10.1145/2639108.2639135.

[25] Qian Wang et al. 'Rain Bar: Robust Application-Driven Visual Communication Using Color Barcodes'. In: *2015 IEEE 35th International Conference on Distributed Computing Systems*. 2015, pp. 537–546. DOI: 10.1109/ICDCS.2015.61.

[26] *What LCD Modes Mean: Reflective, Transmissive, Transflective*. https://www.newvisiondisplay.com/lcd-modes/. 2017.

[27] Wikipedia contributors. *YUV*. 2021. URL: https://en.wikipedia.org/wiki/YUV#Relation_with_Y%E2%80%B2CbCr.

[28] Xieyang Xu et al. 'PassiveVLC: Enabling Practical Visible Light Backscatter Communication for Battery-Free IoT Applications'. In: *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. MobiCom '17. Snowbird, Utah, USA: Association for Computing Machinery, 2017, pp. 180–192. ISBN: 9781450349161. DOI: 10.1145/3117811.3117843. URL: https://doi.org/10.1145/3117811.3117843.

[29] Jackie (Junrui) Yang and James A. Landay. 'InfoLED: Augmenting LED Indicator Lights for Device Positioning and Communication'. In: UIST '19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 175–187. ISBN: 9781450368162. DOI: 10.1145/3332165.3347954. URL: https://doi.org/10.1145/3332165.3347954.

[30] Zhice Yang et al. 'Wearables Can Afford: Light-Weight Indoor Positioning with Visible Light'. In: MobiSys '15. Florence, Italy: Association for Computing Machinery, 2015, pp. 317–330. ISBN: 9781450334945. DOI: 10.1145/2742647.2742648. URL: https://doi.org/10.1145/2742647.2742648.

[31] Kai Zhang et al. 'ChromaCode: A Fully Imperceptible Screen-Camera Communication System'. In: *IEEE Transactions on Mobile Computing* 20.3 (2021), pp. 861–876. DOI: 10.1109/TMC.2019.2956493.

[32] *ZXing*. https://github.com/zxing/zxing. 2020.

List of Acronyms and Abbreviations

AMLED	Active matrix light-emitting diode
DSLR	Digital single-lens reflex camera
FLC	Ferroelectric liquid crystal
FLCoS	Ferroelectric liquid crystal on silicon
FPS	Frames per second
HDMI	High-Definition Multimedia Interface
LC	Liquid crystal
LCD	Liquid-crystal display
LED	Light-emitting diode
NTSC	National Television System Committee
OFDM	Orthogonal frequency-division multiplexing
PBS	Polarizing beam splitter
QR code	Quick Response code
ROI	Region of interest
VLC	Visible light communication

Appendix A

Working principle of the FLCoS micro-display module

Figure A.1 is the schematic drawing of the LV201k micro-display module, whose main components include:

- a tri-color LED,
- a reflective silicon backplane,
- a layer of ferroelectric liquid crystal, and
- a polarizing beam splitter (PBS).

The display module works in the following manner:

1. The LED emits light through a pre-polarizer and diffuser to avoid stray light with unwanted polarization and to create an even illumination.
2. The red, green and blue light is polarized and reflected by a PBS onto the liquid crystal array.
3. The video data is fed to the silicon backplane, controlling the voltage applied to its top reflective and pixelated surface. The voltage determines the orientation of the liquid crystal molecules, which in turn determines the polarization state of the light reflected from the panel.
4. The PBS will only allow light with one type of polarization to pass through, thereby creating bright or dark pixels according to the voltage that was applied to the pixels.

The display module show colors by sequentially emitting light in red, green and blue. The image data is separated into three components corresponding to the colors. The three components of an image are displayed at a very high frequency. Human eyes integrate all the colors and hence see an image with the intended color.

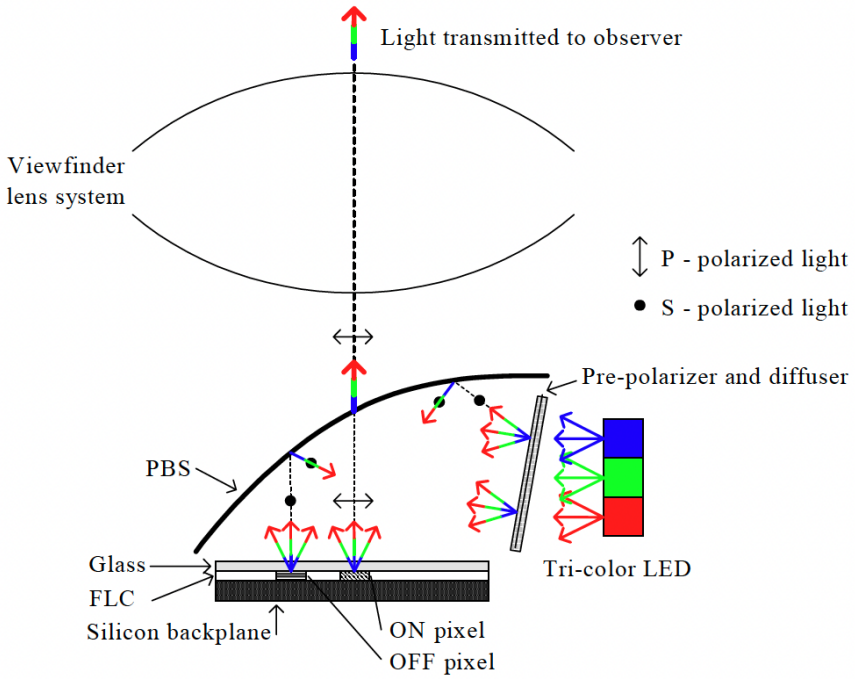


Figure A.1: **Pictorial of LV201k display module [15].**

Appendix B

Data Source Video Generation

For this work, the message for transmission is encoded as two sequences of QR codes, resulting in two video files for the two transmitters. Here is the Python code to generate two separate video files from texts:

```
import os
import qrcode
from PIL import Image
import numpy as np
import cv2
import jnius_config
jnius_config.set_classpath(r'./JavaReedSolomon.jar')
from jnius import autoclass

### Constants for QR version 1
QR_BYTES = 17 # Capacity of each QR in bytes.
RS_DATA_SIZE = 191 # Data capacity of Reed Solomon code.
RS_PARITY_SIZE = 64 # Parity size of Reed Solomon code.
RS_TOTAL_SIZE = 255 # Total size of Reed Solomon code.

### QR code encoder
qr = qrcode.QRCode(
    version=1,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=20,
    border=2,
)

### Read text file in binary mode
file = open(r'./blackcat1.txt', 'rb')
file_bytes = file.read()


### Divide the data into `QR_BYTES` bytes & fill into each row of
    the file_matrix
file_matrix = []
for i in range(RS_DATA_SIZE):
    row = bytearray( file_bytes[ i*(QR_BYTES-1) : i*(QR_BYTES-1)+(
    QR_BYTES-1)] )
    file_matrix.append(row)
```

```
34
35 ### fill the rest of the file_matrix with 0s
36 for i in range(RS_DATA_SIZE,RS_TOTAL_SIZE):
37     row = bytearray([0 for x in range(QR_BYTES-1)])
38     file_matrix.append(row)
39
40 #### Reed Solomon encoder.
41 RS = autoclass('com.backblaze.erasure.ReedSolomon')
42 rs = RS.create(RS_DATA_SIZE, RS_PARITY_SIZE)
43 rs.encodeParity(file_matrix, 0, QR_BYTES-1)
44
45 img_array = []
46 for i in range(RS_TOTAL_SIZE): # The sequence consists of `
       RS_TOTAL_SIZE` QR codes.
47     ### Create data as list of bytes.
48     data = bytes( bytes([i]) + bytes( file_matrix[i][0:QR_BYTES] )
       )
49
50     ### Clear QR data and generate new QR.
51     qr.clear()
52     qr.add_data(data)
53     qr.make()
54     imgqr = qr.make_image(fill_color="black", back_color="white")
55
56     #### Convert QR image to OpenCV.
57     imgnp = np.array(imgqr)
58     imgnp = imgnp.astype(np.uint8)*255
59     img = cv2.cvtColor(imgnp,cv2.COLOR_GRAY2BGR)
60
61     ### Add to image array
62     img_array.append(img)
63
64 ### Dimensions of the video.
65 height, width, layers = img_array[0].shape
66 size = (width, height)
67
68 ### create video file
69 video01 = cv2.VideoWriter('B&W-V1-30FPS-1stHalf.avi',cv2.
       VideoWriter_fourcc(*'DIVX'), 30, size)
70 video02 = cv2.VideoWriter('B&W-V1-30FPS-2ndHalf.avi',cv2.
       VideoWriter_fourcc(*'DIVX'), 30, size)
71
72 ### fill the video(s) frames with QR images
73 half = len(img_array) / 2
74 for i in range(len(img_array)):
75     if i <= half:
76         video01.write(img_array[i])
77     else:
78         video02.write(img_array[i])
79
80 ### Release video files
81 video01.release()
82 video02.release()
```