2017-06

VISUAL LIGHT COMMUNICATION USING IMAGE PROCESSING IN OPENCL

Heinbach, Kathleen A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

VISUAL LIGHT COMMUNICATION USING IMAGE PROCESSING IN OPENCL

by

Kathleen A. Heinbach

June 2017

Thesis Advisor: Weilian Su
Second Reader: Monique P. Fargues

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE VISUAL LIGHT COMMUNICATION USING IMAGE PROCESSING IN OPENCL			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kathleen A. Heinbach			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT

This project explored the use of cameras and image-processing programs to establish a visible light communications link. The system, which consisted of a Raspberry Pi that drove a red-light emitting diode and camera, was connected via transmission control protocol to a graphics-processing unit with an OpenCL image-processing program used to decode the transmission. The system achieved a maximum data transfer rate of 10.0 bits per second with 0.005 bit error ratio with one LED. It achieved a maximum data rate of 20.0 bps with 0.143 bit error ratio with two LEDs. The system performance is limited by the low frame rate of the Raspberry Pi camera. Further improvements could include replacing the camera with a high-speed device to increase the data rate and improving the system's resilience to interference.

14. SUBJECT TERMS visible light communication, OpenCL, image processing, parallel processing			**15. NUMBER OF PAGES** 47
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

VISUAL LIGHT COMMUNICATION USING IMAGE PROCESSING IN OPENCL

Kathleen A. Heinbach
Ensign, United States Navy
B.S., United States Naval Academy, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Weilian Su
Thesis Advisor

Monique P. Fargues
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This project explored the use of cameras and image-processing programs to establish a visible light communications link. The system, which consisted of a Raspberry Pi that drove a red-light emitting diode and camera, was connected via transmission control protocol to a graphics-processing unit with an OpenCL image-processing program used to decode the transmission. The system achieved a maximum data transfer rate of 10.0 bits per second with 0.005 bit error ratio with one LED. It achieved a maximum data rate of 20.0 bps with 0.143 bit error ratio with two LEDs. The system performance is limited by the low frame rate of the Raspberry Pi camera. Further improvements could include replacing the camera with a high-speed device to increase the data rate and improving the system's resilience to interference.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ASK	amplitude shift keying
BER	bit error ratio
CPU	central processing unit
CSI	camera serial interface
FPGA	field programmable gate array
GPIO	general purpose input output
GPGPU	general purpose graphic processing unit
GPU	graphics processing unit
LED	light emitting diode
OOK	on-off keying
PPM	pulse position modulation
RGB	red, green, blue
TCP	transmission control protocol
VLC	visible light communication

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Weilian Su for his support and guidance over the course of my research. I would also like to thank the NPS cycling club for challenging and encouraging me during my time here at NPS. Finally, I would like to thank my friends and family for their constant support throughout the last year.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Wireless communication is essential to everyday civilian and military life. Conventional wireless communication systems operate in the frequency band between 100 MHz and 14.5 GHz. This portion of the electromagnetic spectrum is overcrowded. It is regulated by the International Telecommunications Union on an international scale, and by the Federal Communications Commission within the United States [1]. The frequency range of visible light is between 400 THz and 800 THz, so light does not interfere with existing electromagnetic signals in sensitive areas, such the battleground or hospitals [2]. High-powered radio frequency radiation can also pose health hazards that visible light does not present [3].

In addition, systems such as the Internet-of-Things or networks of small drones require communication capabilities that can be established with low-power transmissions. Light-emitting diodes consume a relatively small amount of power when compared to traditional radio transmitters and can be switched ON and OFF at speeds higher than those that are visible to the human eye, so they are appealing for use in communications [2]. High-power LEDs are also increasingly used for illumination purposes. Therefore, communication signals can be encoded using either high-powered LEDs used for illumination without requiring the expenditure of additional power or small, low-powered LEDs.

Visible light communication (VLC) can also provide a level of security not achievable with other signals in the electromagnetic spectrum. Light cannot travel through opaque barriers. Thus, a VLC network within a room cannot be intercepted by anyone outside of the room, making it an attractive solution for transmitting highly sensitive data over short distances.

B. OBJECTIVES

The first objective of this project was to encode data onto a visible light signal using an LED and on-off keying. The second objective was to sample the visible light

signal with a camera, transmit the sampled frames to a graphical processing unit, and decode the signal using an OpenCL image-processing program. The third objective was to add a second LED to double the data rate of the communications link.

C. THESIS ORGANIZATION

This thesis comprises five chapters. An overview of visible light communication and OpenCL is provided in Chapter II. The hardware configuration and decoding algorithm are described in Chapter III. The performance of the communication link is presented in Chapter IV. Finally, conclusions and recommendations for future research are discussed in Chapter V.

II. BACKGROUND

This chapter introduces visible light communication and discusses existing visible light communication systems. It also describes several signal modulation schemes used in visible communication systems. The red, green and blue (RGB) color model is described, and an overview of the OpenCL programming framework is given. Finally, the OpenCLIPP library is introduced.

A. VISIBLE LIGHT COMMUNICATION

Humans can perceive light with wavelengths ranging from 380 nm to 750 nm [2]. Visible light communication (VLC) is the use of light within this range to transmit information wirelessly. A VLC system consists of an illumination source used to encode data onto a light signal and a sensor used to decode the light signal. The data rate in a VLC system depends on the speed at which the illumination source can be switched between ON and OFF modes, so sources that can flash rapidly ON and OFF are essential.

LEDs and fluorescent lighting are the two most common light sources used in VLC. Although incandescent lighting is a popular source of illumination, it cannot reliably be switched quickly enough to meet the requirements of most VLC applications. Fluorescent lighting and LEDs are appealing for VLC applications because they are both commonly used for illumination. Both can flicker faster than the human eye can perceive, so they can be used simultaneously for VLC and illumination without visibly flickering. In addition, LEDs are quickly replacing both incandescent and fluorescent lights for illumination so they are the most common light source used in VLC applications.

Intensity modulation with direct detection is commonly employed in communication schemes using photodiodes [3]. Photodiodes generate current from incident light that is proportional to the intensity of the received optical wave. The wave phase and frequency information is not used in demodulation. Previous visible light communication systems based on LEDs and photodiodes have used pulse-position modulation (PPM) [4]. In PPM, there are a specified number of slots per symbol and the specific slot determines the symbol decoded value. PPM requires a high level of

complexity because the detector must be synchronized to both the slot and symbol periods.

Another implementation used LEDs as light sensors to create full duplex communication links [5]. Note that LEDs cannot be used to receive and transmit simultaneously, so they must alternate between transmitting and receiving phases. During the transmitting phase, the LED is operated in forward bias mode and outputs light. During the receiving phase, the LED is operated in reverse-biased mode. The LED gets charged at the beginning of the OFF phase, and the difference between the charged voltage and voltage remaining at the end of the OFF phase is used to measure the amount of light received during this phase. The LED-to-LED communication channel used a 2-pulse position modulation scheme with guard intervals. The amount of light leakage from "on" slots into "off" slots was used to maintain synchronization between the two devices.

Cameras can also be used for light detection in VLC. One implementation explored using smartphone cameras and LEDs to create a VLC link [6]. Binary frequency shift keying was employed in this scheme, with the LED blinking at one frequency to transmit a zero bit and at a different frequency to transmit a one bit. In the first decoding method proposed in [6], the average light intensities of successive frames from one bit were stored in a vector and the Fourier transform applied to determine the specific frequency. A second method used the electronic rolling shutter effect to determine the value of the transmitted bit. The electronic rolling shutter effect is a predictable distortion that occurs because smartphone cameras take images using a fixed exposure time. The distortion depends on the blinking frequency of the LED, which can be used to distinguish between the two frequencies used to transmit data.

Amplitude shift keying (ASK) is a signal modulation scheme in which data is represented by changes in the amplitude of the carrier wave. The simplest form of ASK is on-off keying (OOK). As seen in Figure 1, in OOK, the presence of light within the period defined for one symbol, Ts, represents a binary "1," while the absence of light for that period represents a binary "0." In non-return-to-zero OOK, the signal does not return to the OFF state between symbols. At high transmission rates, OOK can be especially susceptible to inter-symbol interference because of non-linear distortion from the LED.

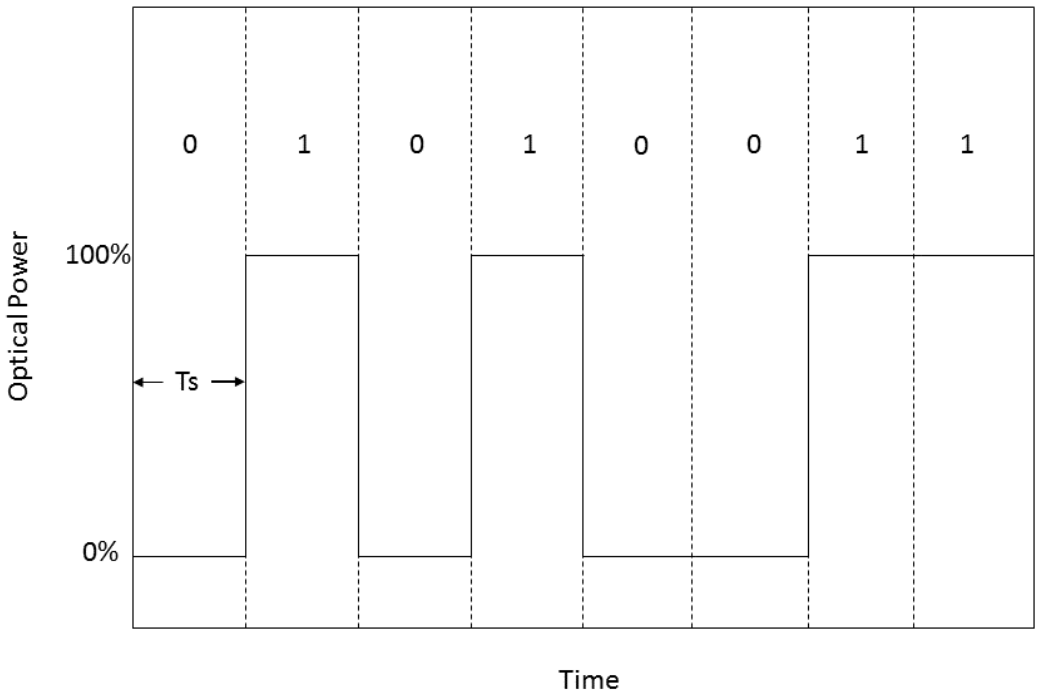


Figure 1. Optical Non-Return-to-Zero On-Off Keying.

B. OPENCL

OpenCL is an open-source parallel programming language for heterogeneous computing platforms [7]. It is portable across computing devices, such as central processing units, graphics processing units, and field programmable gate arrays. Most modern computers are heterogeneous, with a CPU to handle general processing and a GPU to handle graphics processing. Originally, programs written for the CPU and GPU were separate, as the two processing devices served different purposes. In the past couple of decades, researchers have explored using the GPU for tasks other than computer graphics. This lead to the development of general-purpose GPU (GPGPU) programming, which was limited by the lack of a programming language designed for these specific applications.

In 2006, Nvidia released CUDA, a parallel-processing platform that enabled the use of Nvidia GPUs for GPGPU applications [7]. However, CUDA was not well suited for heterogeneous computing, as it could only run on NVidia GPUs. In 2008, OpenCL

emerged as a framework for parallel computing across heterogeneous platforms. OpenCL is suited to run on both GPUs and CPUs manufactured by Intel, AMD, Nvidia, and IBM, as well as embedded devices such as digital signal processors and FPGAs [8]. Programs can be run and synchronized across different processors within a heterogeneous computer simultaneously and share data between processors.

The OpenCL platform model consists of a host computer connected to one or more OpenCL devices [7]. Each device contains a number of compute units, which, in turn, contain a number of processing elements, as seen in Figure 2. The number of compute units contained within a device corresponds to the number of cores in that device. The host controls the host memory, which contains memory for the overall application as well as memory that can be accessed and written to by the individual devices. Data is shared between compute units within the device memory but can be shared only between devices through the host memory.

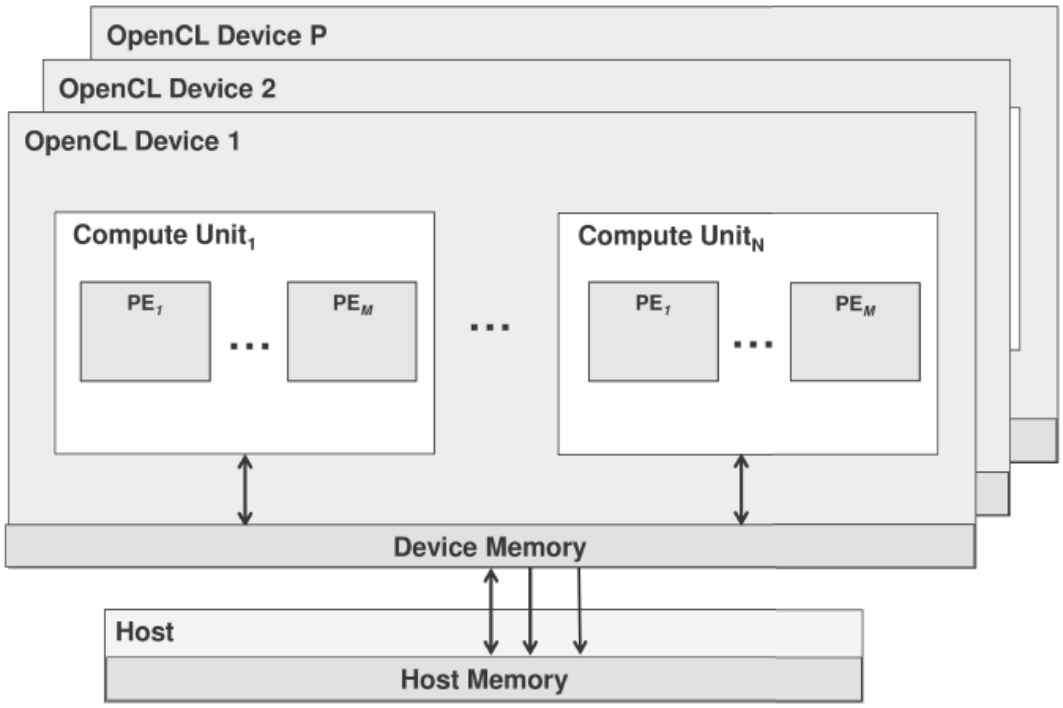


Figure 2. The OpenCL Platform Model. Source: [7].

Each OpenCL program consists of a host application, which controls the overall execution of the program, and one or more kernels, which are the functions executed within the program [8]. The host application is written in either C or C++ code and the kernels are in OpenCL C code. The host application defines the context and parameters such as the devices and memory space that make up the computing environment, and distributes kernels and memory operations into a command queue for each device.

Kernels are further broken down into work-items, which are the individual computations performed within a kernel instance. Work-items that belong to a particular kernel instance are grouped into work-groups. Each work-group gets assigned to a compute unit. Work-items are executed by the processing elements within the compute unit. Vendors decide how their devices are broken down into processing units. The results from each kernel instance are sent back to the host and further processed, as detailed in the host application.

C. OPENCLIPP LIBRARY

The OpenCLIPP library is a library of OpenCL image performance primitives designed for computer vision applications. The library is an alternative to the NVidia and Intel performance primitives. OpenCLIPP provides both C and C++ interfaces for use by developers and researches [9].

The library contains a variety of modules commonly used in image processing. For example, the filter module is capable of applying Gaussian, Sobel, Prewitt, Scharr, Laplacian, and median filters. The arithmetic module contains functions for addition, subtraction, multiplication, and division between two images as well as between an image and a constant value. The blob module detects and labels the edges of connected regions with similar pixel intensity values. The transform module mirrors, flips, transposes, and rotates images. The thresholding module creates binary images from color images and pixel-intensity value threshold. The statistics module computes the maximum, minimum, sum, sum of squares, mean, and standard deviation of the pixel values of an image. The morphology module erodes and dilates images. The logic module performs logical “and”, “or”, “exclusive or”, and “not” operations. The integral module calculates the square

integral sum of the pixel values of an image. The library also contains modules that compute image histograms, fast Fourier transforms, and conversions between color and grayscale formats.

D. RGB COLOR MODEL

One of the most popular ways to represent color in digital images is by combination of red, green, and blue. In the Portable Network Graphics image format used in this project, each color is represented by one byte of data. The possible values for each color range from 0 to 255. As shown in Figure 3, low pixel values correspond to dark colors, while high pixel values correspond to light colors.

Figure 3. Color Range of Red Channel from 0 to 255, from Left to Right. Source: [10].

E. TCP/IP SOCKET PROGRAMMING

In the Transmission Control Protocol (TCP), a socket is a combination of an Internet Protocol (IP) address and port number; it is used to establish a connection between applications running on two separate machines within a network [11]. As shown in Figure 4, the server creates a socket by binding an IP address to a particular port number. It then listens on that port for a client to initiate a connection. Once the connection is established, both the client and server can send and receive data. Finally, the client terminates the connection and the socket closes when the connection is no longer needed.

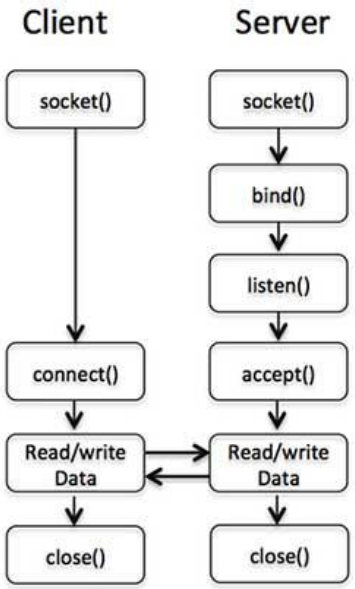


Figure 4. TCP Socket Connection Process. Source: [12].

Recent developments in VLC and described signal modulations used in current VLC implementations were introduced in this chapter. The OpenCL language, the OpenCLIPP library, and the concept of RGB image representation were discussed, as well as the TCP socket connection process concept introduced. The VLC system implemented in this project is discussed next

THIS PAGE INTENTIONALLY LEFT BLANK

III. EXPERIMENTAL SETUP

The hardware configuration and programs used in this project are introduced in this chapter. The hardware components and connections between devices are described. Last, the programs used to drive the LEDs, operate the camera, and decode the received signal are discussed.

A. OVERVIEW

The hardware setup used in this work includes a Super Micro GPU SuperWorkstation 7047 GR-TRF, Raspberry Pi, Raspberry Pi camera module, and two 3.0 mm red LEDs. The Raspberry Pi is connected to the server by an Ethernet cable, as shown in Figure 5. The LED is placed 12.0 cm away from the camera.

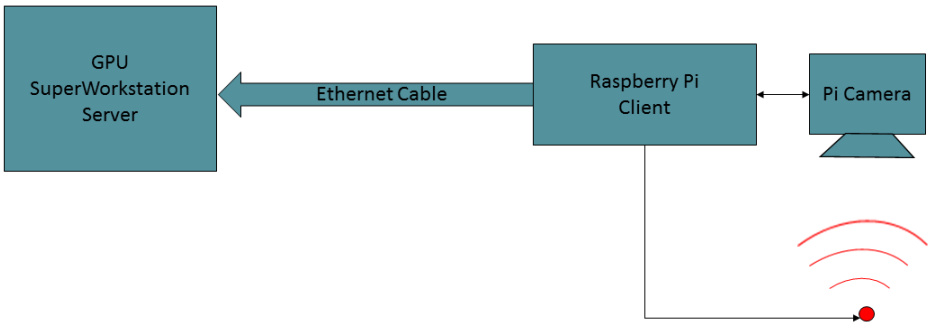


Figure 5. Hardware Configuration Diagram.

B. SUPERMICRO GPU SUPERWORKSTATION 7047GR-TRF

The SuperMicro GPU SuperWorkstation 7047GR-TRF is a tower or rack mountable server with four double-width slots for graphics cards and two Gigabit Ethernet LAN ports [13]. It is equipped with an Intel Xeon Processor E5-2643 CPU. The CPU contains four cores and eight threads [14], and operates at a base frequency of 3.30 GHz and a maximum frequency of 3.50 GHz. The server is also equipped with two Nvidia Tesla K20c graphics cards and one Nvidia GeForce GTX 650 graphics card. The

Tesla K20c GPU contains 2496 processor cores and operates at a core frequency of 706 MHz [15]. The GeForce GTX 650 contains 384 processor cores and operates at a base frequency of 1058 MHz. The CPU and all three of the GPUs in the workstation run OpenCL version 1.2. Image processing operations implemented for the visual communications channel are performed on the CPU and GPUs in the workstation. The workstation acts as a server in establishing a TCP socket for receiving images from the Raspberry Pi via one of the Ethernet LAN ports.

C. RASPBERRY PI

The Raspberry Pi 3 is a small, cheap single-board computer that is popular for electronics hobbyists and widely used in schools. It comes equipped with a 1.2 GHz quad core ARMv8 CPU, 40 general purpose input-output pins, an Ethernet port, and camera serial interface (CSI) for the Raspberry Pi camera module [16].

The Raspberry Pi configuration used in this project is shown in Figure 6. The two LEDs used to transmit the signal were driven using the GPIO pins. The camera was connected to the CSI, and the Ethernet port was used to connect to the workstation. The LEDs were placed 12.0 centimeters away from the camera. All programs used on the Raspberry Pi were Python scripts.

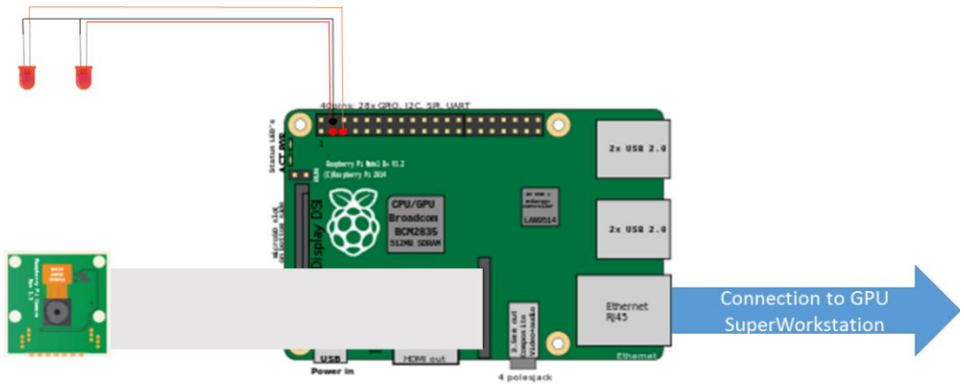


Figure 6. Raspberry Pi Configuration.

1. LED Driver

The LED driver program started with a calibration period for the image-processing program. As seen in Figure 7, during phase 1, the LED is ON for two bit periods. This phase is used to calibrate the pixel value threshold in the decoding program. Then, in phase 2, the LED flashes ON and OFF four times, remaining in each state for one period. The decoding program uses the frames from that phase to set the pixel count threshold. The final transition from ON to OFF, in phase 3, indicates the decoding program should process the following transmission as data. Data is transmitted in phase 4. For our testing purposes, the data is randomly generated; the program goes into a loop where it randomly chooses a one or a zero, with equal probabilities. Afterwards, the bit value is written to a text file to check against the decoded result. If the bit value is a one, the LED is ON for one period; if it is a zero, the LED is OFF for one period. After a pre-defined number of data bits (100 bits in Figure 7) has been transmitted in phase 4, the LED flashes ON and OFF, repeating phase 3, to resynchronize with the image-processing program; afterwards, phase 4 is repeated, where data is transmitted.

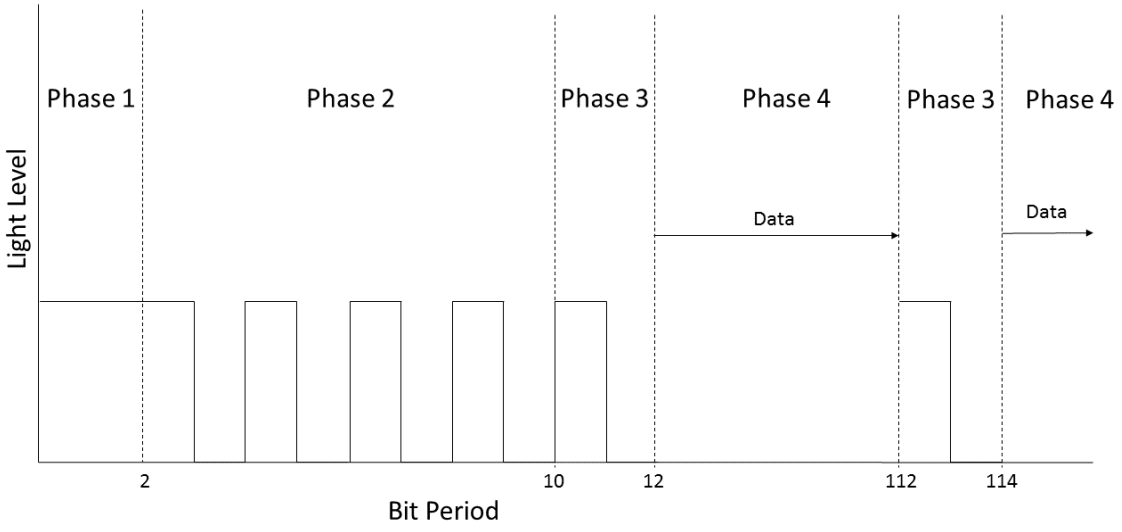


Figure 7. Initialization and Data Transfer Processes.

2. Raspberry Pi Camera

The camera program is a modified version of the "rapid capture and streaming" example in the Raspberry Pi Camera Module documentation [17]. The program, acting as a client, establishes a TCP connection with the workstation. It then simultaneously captures images and transmits them across the wired connection. The resolution was lowered to 100 pixels x 100 pixels to reduce the image transfer time. The camera's ISO and white balance settings were set to constant values to prevent the camera from automatically adjusting the settings as the lighting conditions changed.

D. DECODING ALGORITHM

The decoding algorithm implemented at the server begins by calibrating the two threshold values used to evaluate the status of the LED in order to extract the signal transmitted by the blinking of the LED. Note that only the red channel from the RGB pixel values is used in the decoding process, because the LEDs used are red. Multiple frames are used to determine each bit value. The number of frames per bit is the frame rate of the camera multiplied by the period of one bit *Ts* and is calculated as :

$$Frames\ per\ bit\ = Frame\ rate * T_s\ . \quad (1)$$

For example, when the data rate is 10 bits per second and the frame rate is 30 frames per second, we have 3 frames per bit. The first threshold value is the pixel value threshold, which is calculated and set during phase 1. The frames of the first two-bit periods are used to calculate and set the pixel value threshold. During this period, the LED is ON, as seen in Figure 7. The thresholding function from the OpenCLIPP library is performed repeatedly on the first frame, with the pixel value threshold starting at zero and incrementing until fewer than half of the pixels in the frame have red values over the pixel value threshold. The 50% mark was chosen via visual inspection of results obtained from thresholding frames containing ON LEDs with several different thresholds.

The process is repeated on the remaining frames of the first two bit periods, with the pixel value threshold increasing again if more than half of the pixels in the frame have values above the pixel value threshold from the previous frame. At the end of this process, the pixel value threshold is the lowest value for which fewer than half of the

pixels in each frame from phase 1 have higher values. The resulting five frames obtained from the thresholding function and the calculated pixel value threshold are shown in Figure 8.

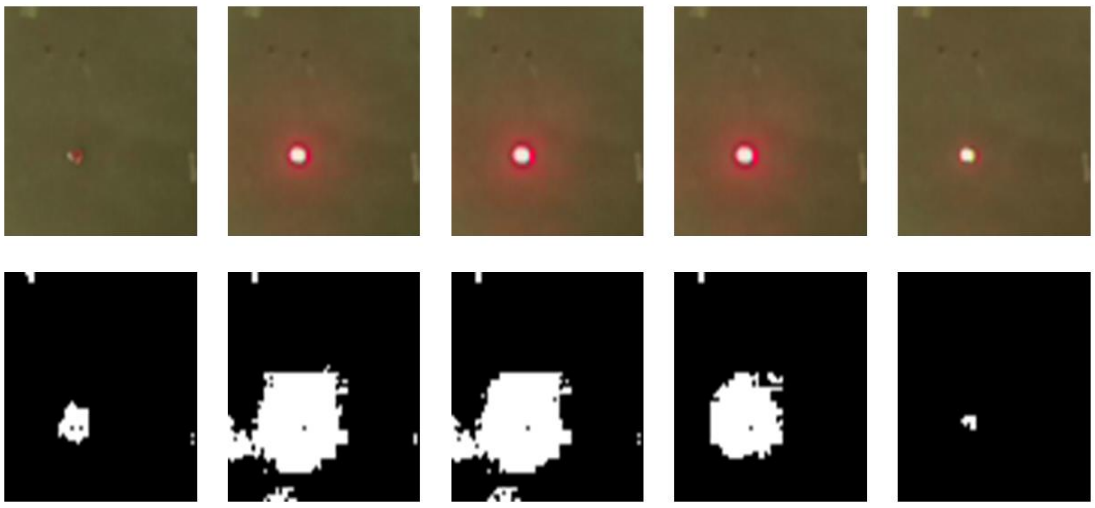


Figure 8. Original Frames (Top Row) and Frames Following Thresholding (Bottom Row).

The second threshold value determined in the calibration process is the pixel count threshold. The pixel count is the number of pixels in a frame that have values higher than the pixel value threshold. The pixel count threshold value is the pixel count value above which a frame is determined to contain an ON LED and below which a frame is determined to contain an OFF LED. During the eight-bit periods of phase 2, the LED flashes ON and OFF four times, each for the duration of one bit period. The total number of pixels with values equal to or higher than the pixel value threshold in each frame is the pixel count for the frame. The maximum pixel count and the minimum pixel count collected in phase 3 are averaged to determine the pixel count threshold. An example of pixel counts from phase 2 and the calculated pixel count threshold is shown in Figure 9. The pixel count (in blue) and pixel count threshold (in green), for a transmitted sequence of alternating ones and zeroes are shown in Figures 10 and 11 respectively. Figure 11 is a closer view of 13 bits extracted from the same sequence.

When the program starts processing data frames, the LED is labeled ON for a frame when the pixel count for the frame is higher than the pixel count threshold value, while the LED is labeled OFF for that frame when the pixel count is lower than the pixel count threshold value.

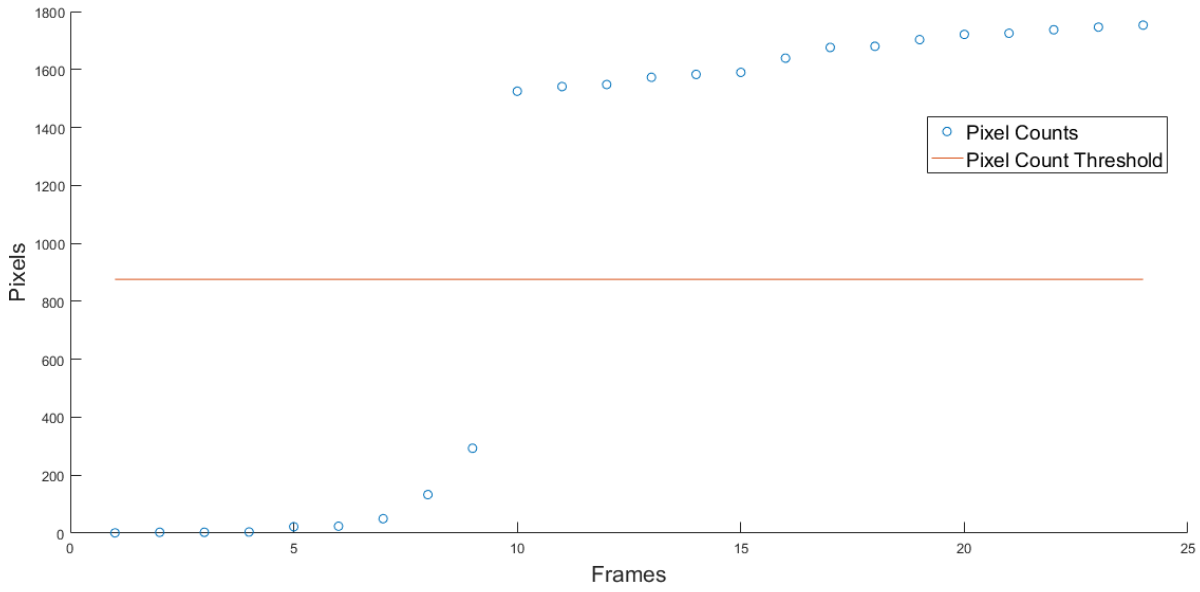


Figure 9. Phase II Pixel Counts and Calculated Pixel Count Threshold.

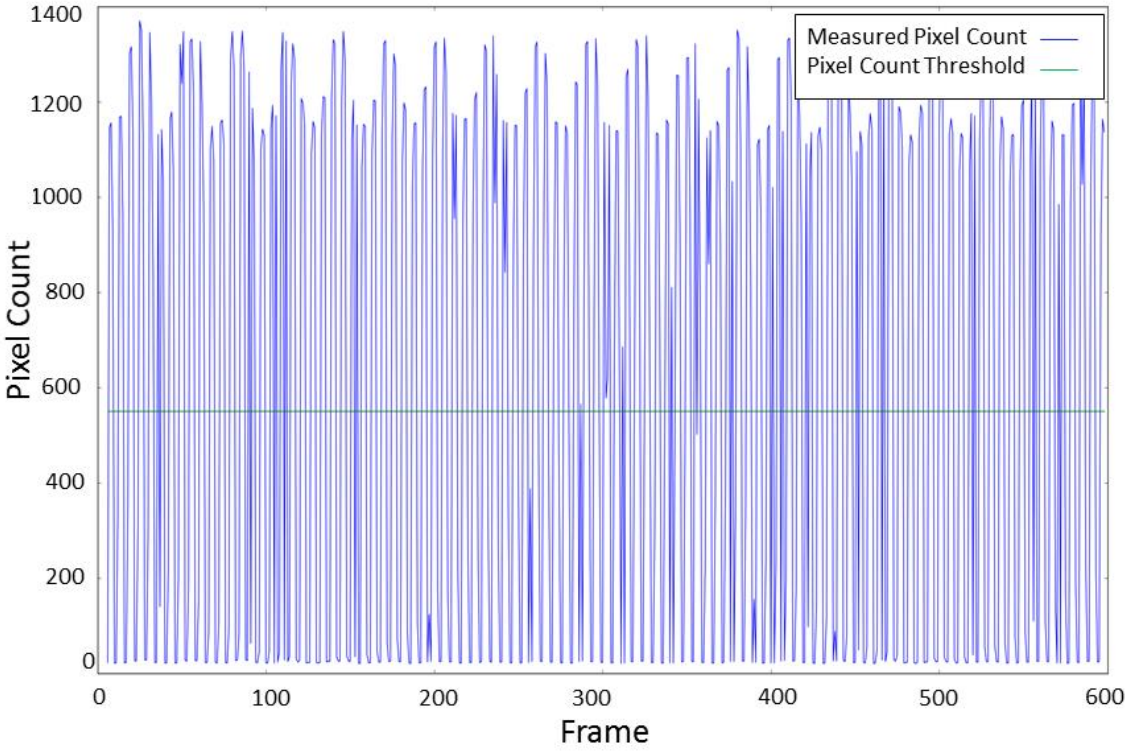


Figure 10. Pixel Counts for an Alternating Sequence of Ones and Zeros.

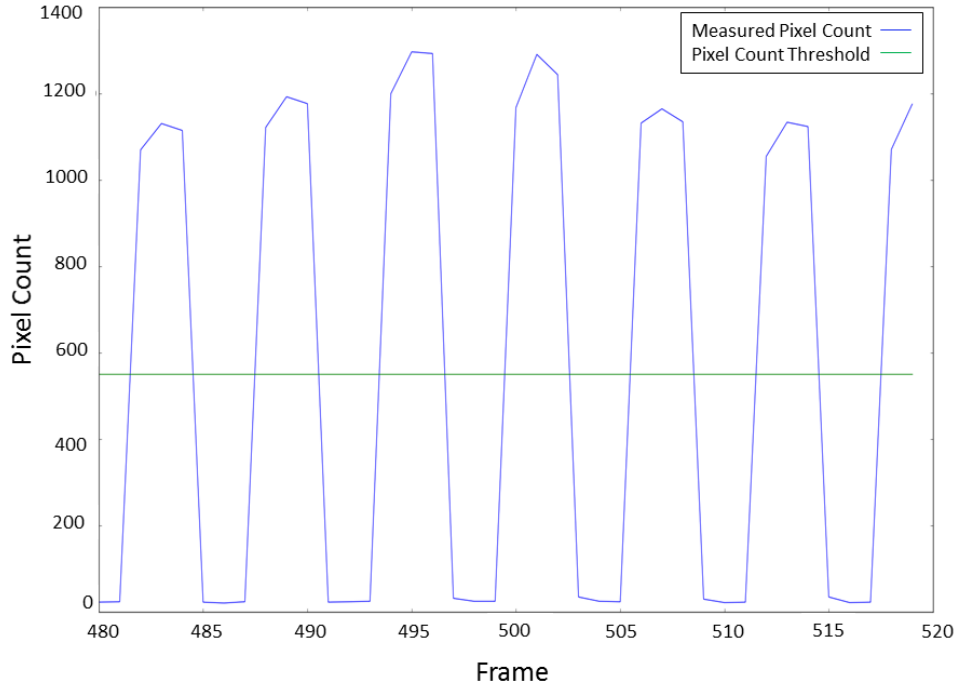


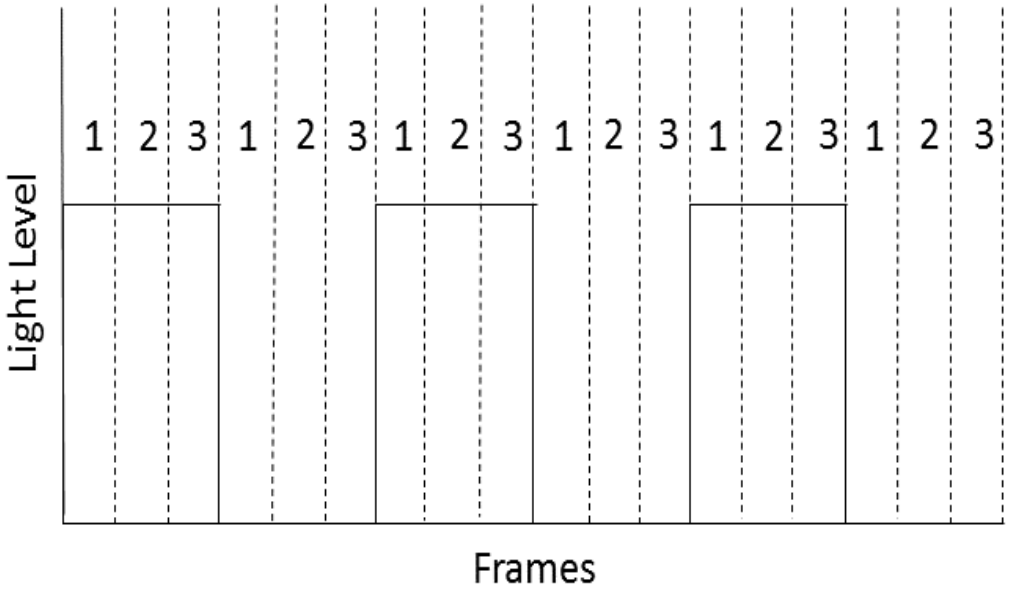
Figure 11. Closer View of Sequence in Figure 8.

Once both thresholds have been calculated, the decoding program waits until the next transition of the LED from ON to OFF to begin decoding the incoming bit stream. At that point, the program assigns each frame a value of one or zero based on whether the LED is determined to be ON or OFF in that frame. Next, the frame values representing a bit are summed. The bit is determined to be a one when the total obtained is greater than one half the number of frames per bit, and determined to be a zero otherwise. Finally, the bit value is then written to a text file.

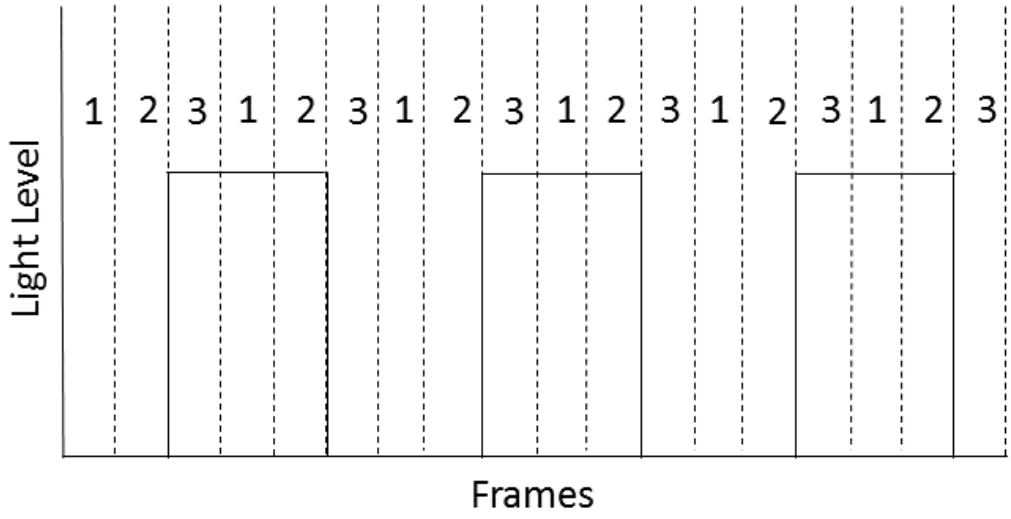
For every 100 bits in phase 4, the program waits for the LED to transition from ON to OFF (re-entering phase 3, as shown in Figure 7) before resuming decoding; this step ensures the decoding program is re-synchronized with the LED transmitter. Such a step is necessary to account for the variations in frame rate and transmission speed across the Ethernet link, as eventually the decoding program lags behind the LED pulses so that the program incorrectly groups frames from different bits together. An example of frames grouped correctly to decode a series of bits, with three frames per bit, and frames grouped incorrectly as the lag accumulates is shown in Figure 12.

A random binary sequence with the pixel count values in blue, the pixel count threshold in green, and the data sequence, as determined by the decoding program, is shown in Figure 13. No errors occurred in the decoded sequence.

Next, a second LED was placed four centimeters away from the first LED to double the data rate. The processing algorithm divides the frame in half, which adequately separates the two LED pulses, as shown in Figure 14. The two half frames are processed following the same method as that used for the full frame with a single LED.



a) Correct



b) Incorrect

Figure 12. Correct Frame Grouping and Incorrect Frame Grouping.

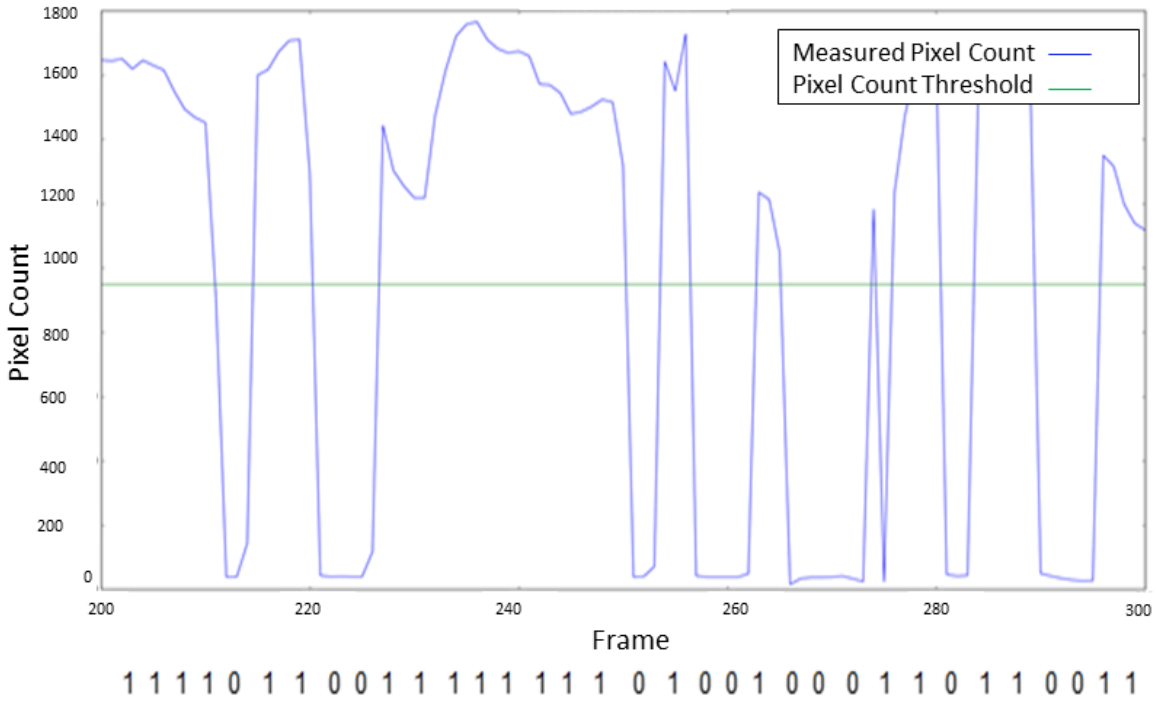


Figure 13. Pixel Count Values and Output Bits for Random Binary Sequence.

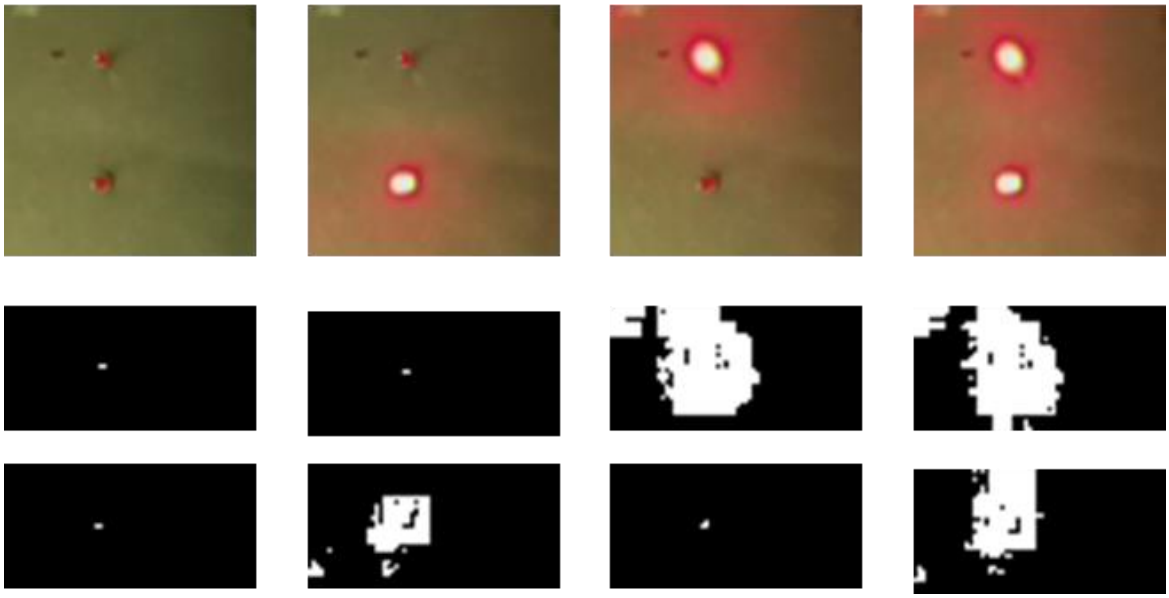


Figure 14. Original Frames before Pixel Value Thresholding (Top Row) and Divided Frames after Pixel Value Thresholding (Bottom Two Rows).

The hardware configuration and programs used to create a VLC link were presented in this chapter. The server, Raspberry Pi, Raspberry Pi, and connections between the components were described as well as the programs used to transmit data using the blinking of the LED, operate the camera, and decode the received frame. Results are discussed in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PERFORMANCE EVALUATION

The tests performed using the communication system and results obtained are described in this chapter. The system was evaluated using one and two LEDs, several data rates, and two different synchronization intervals. For each data rate and synchronization interval considered, an average bit error ratio obtained after twenty test runs is calculated with each test running for 115 seconds.

A. ONE LED

The first trials were performed using one LED. The maximum data rate at which the algorithm can accurately determine the state of the LED is ten bits per second (bps). The maximum data rate of the Raspberry Pi camera is thirty frames per second. In order for the decoding algorithm to stay synchronized with the LED, there must be an integer number of frames per bit. In addition, there must be more than 2 frames per bit for the decoding program to make a clear decision based on a majority rule.

Note that there are three frames per bit, leading to three evaluations of the LED state, when the LED transmits ten bits per second. These three initial evaluations are then combined and the final bit value decision (one or zero) made using the majority rule. However, when the data rate is increased to 15 bits per second, resulting in two frames per bit, the majority rule cannot be applied to make a decision on the bit value. Therefore, the maximum data rate tested on the communication link was 10 bps.

Tests show the bit error ratio of the communication link increases when the data rate increases, as illustrated in Table 1. This BER increase was expected, as there are half as many frames used to determine the value of a bit when the data rate increases from 5 bps to 10 bps. The synchronization interval is defined as the number of data symbols transmitted in phase 4. With one LED, there is one bit per symbol, while with two LEDs, there are two bits per symbol. Results also show the BER increases when the interval between synchronization phases (referred to as phase 3 in Figure 7) increases. This result was expected because it takes longer to correct errors due to synchronization drifts as the interval increases.

Table 1. Communication Link Performance with One LED.

Data Rate	Bit Error Ratio	Synchronization Interval
10 bps	0.00548	50 symbols
10 bps	0.01651505	100 symbols
20 bps	0.011917	50 symbols
20 bps	0.03113195	100 symbols

B. TWO LEDS

The addition of a second LED consistently increases the BER of the communication link, as shown in Table 2. The BER also increases as the data rate increases, as it did with one LED.

Table 2. Communication Link Performance with Two LEDs.

Data Rate	Bit Error Ratio	Synchronization Interval
5 bps	0.02393525	50 symbols
5 bps	0.041904575	100 symbols
10 bps	0.07012285	50 symbols
10 bps	0.1427399	100 symbols

The results of the experiments conducted in the study were discussed in this section. Results show that increasing the number of LEDs, the data rate, and the interval between synchronization phases all increased the BER. Conclusions and recommendations for future work are presented next.

V. CONCLUSIONS AND FUTURE WORK

In this chapter, we present conclusions and recommendations for future work.

A. CONCLUSIONS

In this study, a visual light communication system was implemented using LEDs, a Raspberry Pi Camera, and an OpenCL decoding program. The LED transmitted data using an OOK signal modulation scheme. The decoding program automatically set thresholds for both the value of a pixel to be considered part of the LED pulse and the number of pixels above that threshold in a frame in order for the LED to be considered ON in that frame. The program counted ON and OFF frames to decode the transmitted bits.

The communication link achieved the lowest BER of 0.00548 when one LED was transmitting at 5 bps and the program resynchronized every 50 bits. Adding a second LED, increasing the data rate, and increasing the length of time between synchronization phases also increased the system BER. The maximum data rate achieved was 20 bps obtained when using two LEDs with a resulting BER equal to 0.1427.

B. RECOMMENDATIONS FOR FUTURE WORK

In the future, integrating a high-speed camera with a significantly higher frame rate would dramatically improve the data rate. Adding more LEDs or a panel of LEDs could also offer a significant increase in the data rate. The hardware setup could also be used with other functions available in the OpenCLIPP library to fulfill the requirements of other high-speed image-processing applications.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] A. F. Molisch, "Technical challenges of wireless communications," in *Wireless Communications*, 1st ed. West Sussex, United Kingdom: Wiley-IEEE Press, 2011, pp. 27–36. doi: 10.1002/9781119992806.ch2

[2] S. Arnon, "Introduction" in *Visible Light Communication*. Cambridge: Cambridge University Press, 2015.

[3] F. Khan et al., "Applications, limitations, and improvements in visible light communication systems," in *International Conference on Connected Vehicles and Expo*, Shenzhen, 2015, pp. 259–262. doi: 10.1109/ICCVE.2015.46

[4] H. Elgala et al., "Indoor optical wireless communication: potential and state-of-the-art," *IEEE Communications Magazine*, vol. 49, no. 9, pp. 56–62, Sep. 2011. doi: 10.1109/MCOM.2011.6011734

[5] S. Schmid et al., "Continuous synchronization for LED-to-LED visible light communication networks," in *3rd International Workshop in Optical Wireless Communications*, Funchal, 2014, pp. 45–49. doi: 10.1109/IWOW.2014.6950774

[6] G. Corbellini et al., "Connecting networks of toys and smartphones with visible light communication," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 72–78, Jul. 2014. doi: 10.1109/MCOM.2014.6852086

[7] P. Balaji, "OpenCL: the Open Computing Language," in *Programming Models for Parallel Computing*, 1st ed. Cambridge: MIT Press, 2015, pp. 399-428.

[8] M. Scarpino. (2011, August 3). "A gentle introduction to OpenCL," Dr. Dobb's [Online]. Available: http://www.drdobbs.com/parallel/a-gentle-introduction-to-opencl/231002854. [Accessed: Apr. 20, 2017].

[9] M. Akhloufi and A. Campagna, "OpenCLIPP: OpenCL Integrated Performance Primitives library for computer vision applications," in *SPIE Electronic Imaging 2014, Intelligent Robots and Computer Vision XXXI: Algorithms and Techniques*, San Francisco, CA, 2014. pp. 9025–31

[10] "#000000 To #Ff0000 Gradient Color" *ColorHexa*. [Online]. Available: http://www.colorhexa.com/000000-to-ff0000. [Accessed Apr. 24, 2017].

[11] P. Kryzyzanowski, "Introduction to Sockets Programming," *CS 417 Documents*. [Online]. Available: https://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/. [Accessed: Apr. 24, 2017].

[12] "Perl Socket Programming," *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/perl/perl_socket_programming.htm. [Accessed: Apr. 24, 2017].

[13] "GPU SuperWorkstation 7047GR-TRF," Super Micro Computer, Inc. [Online]. Available: https://www.supermicro.com/products/system/4u/7047/sys-7047gr-trf.cfm. [Accessed: Apr. 25, 2017].

[14] "Intel Xeon processor E5-2643 (10M Cache, 3.30 GHz, 8.00 GT/s Intel QPI) product specifications," Intel ARK (Product Specs). [Online]. Available: http://ark.intel.com/products/64587/Intel-Xeon-Processor-E5-2643-10M-Cache-3_30-GHz-8_00-GTs-Intel-QPI. [Accessed: Apr. 25, 2017].

[15] "NVIDIA Tesla K20c," ***TechPowerUp*. [**Online]. Available: https://www.techpowerup.com/gpudb/564/tesla-k20c. [Accessed: Apr. 25, 2017].

[16] "GeForce GTX 650 Specifications," *GeForce*. [Online]. Available: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-650/specifications. [Accessed: Apr. 25, 2017].

[17] "Raspberry Pi 3 Model B," Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: Apr. 27, 2017].

[18] "Advanced recipes," Picamera 1.10 documentation. [Online]. Available: http://picamera.readthedocs.io/en/release-1.10/recipes2.html#rapid-capture-and-streaming. [Accessed: Apr. 27, 2017].

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California