TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

THIẾT KẾ PHẦN MỀM TRÊN THIẾT BỊ DI ĐỘNG SỬ DỤNG TRUYỀN THÔNG BẰNG ÁNH SÁNG NHÌN THẤY ĐƯỢC (VISIBLE LIGHT COMMUNICATION) CHO ỨNG DỤNG ĐỊNH VỊ TRONG NHÀ

NGUYỄN ĐÌNH QUÂN
Quan.nd192034@sis.hust.edu.vn

Ngành KT Điều khiển & Tự động hóa

Giảng viên hướng dẫn: TS. Nguyễn Hoàng Nam

Chữ ký của GVHD

Khoa: Tự động hóa
Trường: Điện – Điện tử

HÀ NỘI, 8/2023

BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐH BÁCH KHOA HÀ NỘI

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ
ĐỒ ÁN TỐT NGHIỆP

Họ và tên sinh viên: Nguyễn Đình Quân

Khóa: 64 Trường: Điện – Điện tử Ngành: KTĐK &TĐH

1. *Tên đề tài:*

Thiết kế phần mềm trên thiết bị di động sử dụng truyền thông bằng ánh sáng nhìn thấy được (VLC) cho ứng dụng định vị trong nhà.

2. *Nội dung đề tài:*

a. Các số liệu ban đầu:

- Phần cứng (đã được thiết kế) gồm:
  - đèn LED tròn có ∅10
  - cụm đèn 4 đèn LED âm tường
  - Đèn LED panel 300 x 1200 x 10 mm
  - Camera trên điện thoại di động Google Pixel 4
- Phần mềm gồm:
  - Android studio
  - Ngôn ngữ Python, Kotlin

b. Các nội dung tính toán, thiết kế phần mềm:

- Nghiên cứu, tìm hiểu về truyền thông bằng ánh sáng nhìn thấy được (VLC).
- Ứng dụng VLC cho định vị trong nhà.
- Nghiên cứu, tìm hiểu về thị giác máy trên điện thoại di động.
- Nghiên cứu về mối quan hệ giữa các thông số trên camera điện thoại di động tới truyền thông bằng ánh sáng nhìn thấy được.
- Thực hiện lắp đặt, đo đạc, lựa chọn khoảng cách tâm phù hợp cho việc lắp đặt đèn trần nhằm mục đích truyền tin.
- Thiết kế phần mềm phần phát:
  - Truyền dữ liệu với tần số và dữ liệu thay đổi được
- Thiết kế phần mềm phần thu:
  - Sử dụng ngôn ngữ Python, Kotlin và các thư viện xử lý ảnh như OpenCV, Numpy, Scipy… để phát triển phần mềm trên thiết bị di động.

- Thay đổi thông số camera trên điện thoại di động để loại bỏ vùng không phải nguồn sáng.
- Xử lý real – time từng ảnh nhận được từ camera
- Tìm ngưỡng tự động để chuyển đổi ảnh thành ảnh nhị phân
- Thuật toán tìm RoI cho 4 đèn LED tròn và đèn LED panel
- Áp dụng Curve fit để xử lý dữ liệu đường RoI đi qua thành bit 0 và 1.
- Phân tích số liệu đưa ra dữ liệu truyền từ phần phát.
- Tính toán tỉ lệ truyền chính xác trong quá trình truyền.

3. *Thời gian giao đề tài:* 04/2023
4. *Thời gian hoàn thành:* 08/2023

Ngày 03 tháng 08 năm 2023

CÁN BỘ HƯỚNG DẪN

Nguyễn Hoàng Nam.

Lời cảm ơn

Em xin gửi lời tri ân sâu sắc tới mọi người đã đồng hành và hỗ trợ em trong suốt quá trình thực hiện đồ án tốt nghiệp. Trước tiên, em muốn bày tỏ lòng biết ơn chân thành đến TS. Nguyễn Hoàng Nam - một giảng viên đầy tâm huyết thuộc Khoa Kỹ thuật Đo và Tin học Công nghiệp. Thầy đã vô cùng tận tâm hướng dẫn và giúp đỡ em trong quá trình nghiên cứu. Bên cạnh đó, em cũng gửi lời cảm ơn đến gia đình em, những người bạn và các thành viên trong IOTeam Lab đã hỗ trợ em rất nhiều trong thời gian vừa qua. Cuối cùng, em xin cảm ơn các thầy, cô và nhà trường đã tạo điều kiện tốt nhất cho em học tập, nghiên cứu trong 4 năm học vừa qua.

Trong quá trình nghiên cứu, do thời gian còn hạn chế và kiến thức còn hạn chế nên nghiên cứu này có thể chưa hoàn toàn đạt yêu cầu mặc dù em đã cố gắng hết sức. Vì vậy, em sẵn sàng tiếp thu mọi ý kiến phê bình và góp ý để bài báo cáo được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Tóm tắt nội dung đồ án

Nội dung thực hiện:

- Tìm hiểu về công nghệ truyền thông bằng ánh sáng nhìn thấy.
- Thiết kế hệ thống truyền thông bằng ánh sáng nhìn thấy
- Các thuộc tính trên camera điện thoại ảnh hưởng đến quá trình giải mã.
- Lập trình phần mềm trên thiết bị di động chạy hệ điều hành Android
- Xử lý ảnh tìm tâm nguồn sáng, đường RoI và xây dựng thuật toán xử lý với nhiều đèn LED.
- Đo tỉ lệ truyền nhận chính xác với các trường hợp.

Công cụ, phần mềm sử dụng:

- Cụm LED trần 4 bóng và đèn LED panel.
- Phần mềm Android studio, Visual Studio Code. Ngôn ngữ Python, Kotlin.

Kết quả:

- Thiết kế hệ thống đo sai số cho mục đích thực nghiệm, bao gồm:
    - Phần phát: gửi chuỗi bits được mã hóa với các tần số
    - Phần thu: thu nhận, xử lý trên phần mềm di động và đưa ra sai số.

Định hướng phát triển:

- Tối ưu thuật toán để giảm thời gian xử lý. Nâng cao thuật toán tự động chỉnh ngưỡng, thực nghiệm với nhiều trường hợp khác nhau.

Sinh viên thực hiện
(Ký và ghi rõ họ tên)

Nguyễn Đình Quân

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

VLC	Visible light communication
LED	Light Emitting Diode
MAC	Media Access Control
OOK	On - off Keying
VPPM	Variable pulse position modulation
CSK	Color Shift Keying
LiFi	Light Fidelity
CMOS	Complementary Metal Oxide Semiconductor
RGB	Red, Green, Blue
CDR	Clock and Data Recovery
AMP	Amplifier
IS	Image Sensor
PD	Photodiode
CCD	Charge Coupled Device
ADC	Analog to Digital converter
GS	Global Shutter
RS	Rolling Shutter
ID	Identification
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
Wi-Fi	Wireless Fidelity
RF	Radio frequency
NRZ	Non-Return-to-Zero
FEC	Forward Error Correction
Ab	Asynchronous Bit
RLL	Run length limited

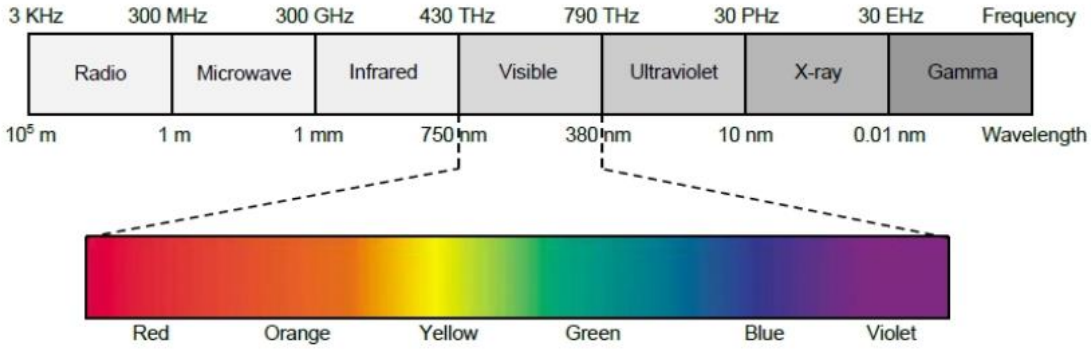
DS	Data Subpacket
PWM	Pulse Width Modulation
RoI	Region of Interest
Tx	Transmitter
Rx	Receiver
Fps	Frame per second

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Giới thiệu chung về VLC

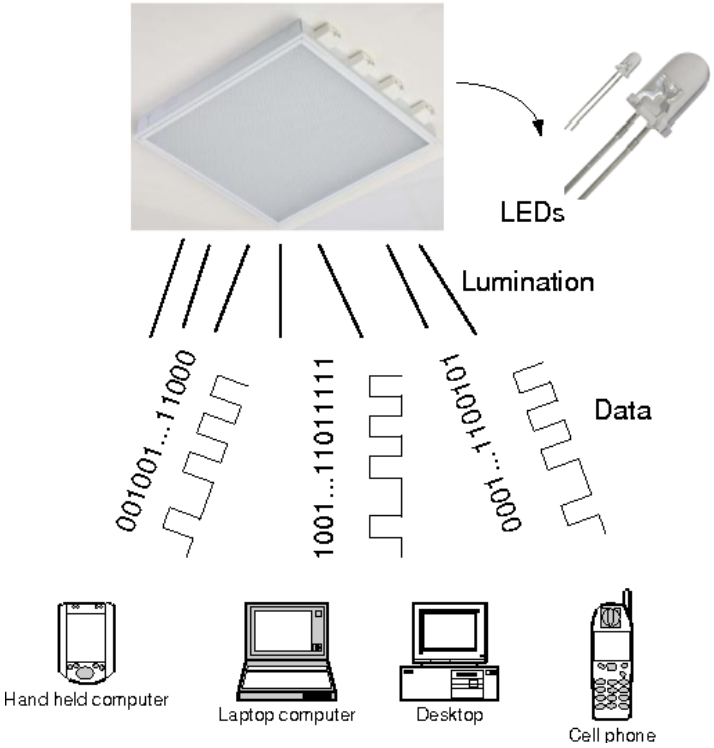
1.1.1 Định nghĩa về VLC

Trong những năm gần đây, một trong những ý tưởng được đưa ra cho việc truyền thông quang không dây là phương pháp truyền thông bằng ánh sáng nhìn thấy được hay Visible Light Communication (VLC). Các tín hiệu trong khoảng sóng từ 380-780 nm [1] của phổ điện từ tương ứng với phổ tần số 430 THz đến 790 THz (Hình 1. 1) là các tín hiệu ánh sáng có thể được mắt người nhìn thấy được. Bằng cách sử dụng các đèn LED, thiết bị chiếu sáng đang rất phổ biến, có thể đạt được cả chiếu sáng và truyền dữ liệu đồng thời. Nhờ đó, việc chiếu sáng bên trong phòng và truyền dữ liệu sẽ được thực hiện mà không cần sử dụng một hệ thống truyền thông khác để bổ sung.



Hình 1. 1 Phổ điện từ

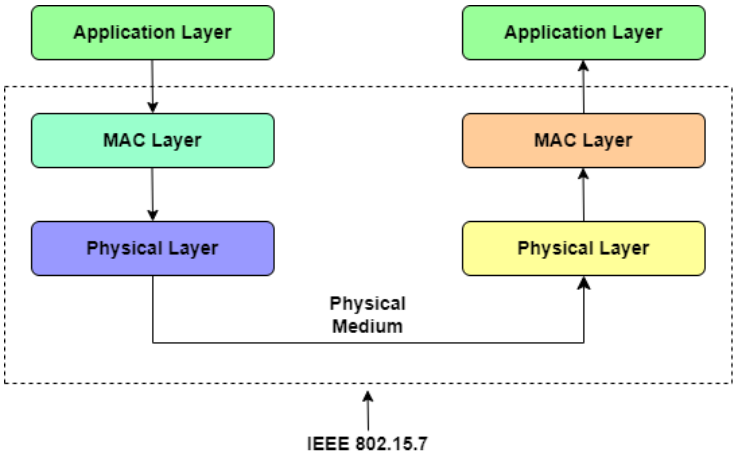
Một hệ thống VLC cơ bản được thể hiện ở Hình 1. 2:



Hình 1. 2 Hệ thống cơ bản VLC.

1.1.2 Kiến trúc của VLC

Hệ thống VLC có bởi 2 phần chính: phần phát và phần thu, thưởng bao gồm ba lớp chung. Chúng là lớp vật lý, lớp MAC và lớp ứng dụng. Kiến trúc lớp của VLC được thể hiện trong Hình 1. 3 [1]. Trong tiêu chuẩn IEEE802.15.7, chỉ có 2 lớp (PHY và MAC) được xác định để đơn giản hóa cho phân loại [2].



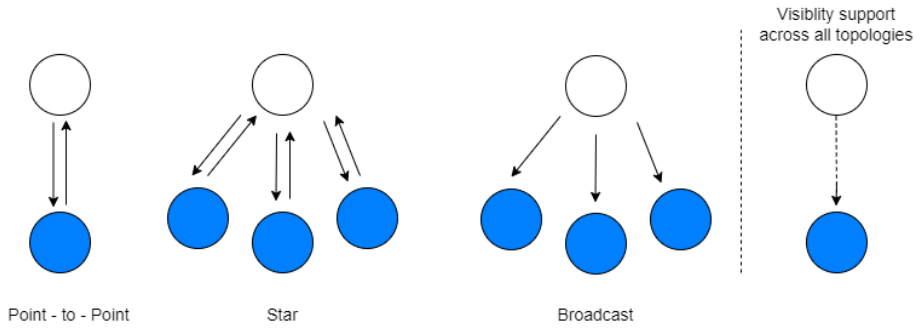
Hình 1. 3 Kiến trúc lớp VLC

a. Lớp MAC

Các tác vụ được thực hiện bởi lớp MAC bao gồm [3]:

- Hỗ trợ di động
- Hỗ trợ điều chỉnh độ sáng
- Hỗ trợ hiển thị
- Hỗ trợ bảo mật
- Các phương pháp giảm hiện tượng nhấp nháy
- Cung cấp một liên kết đáng tin cậy giữa các thực thể MAC ngang hàng.

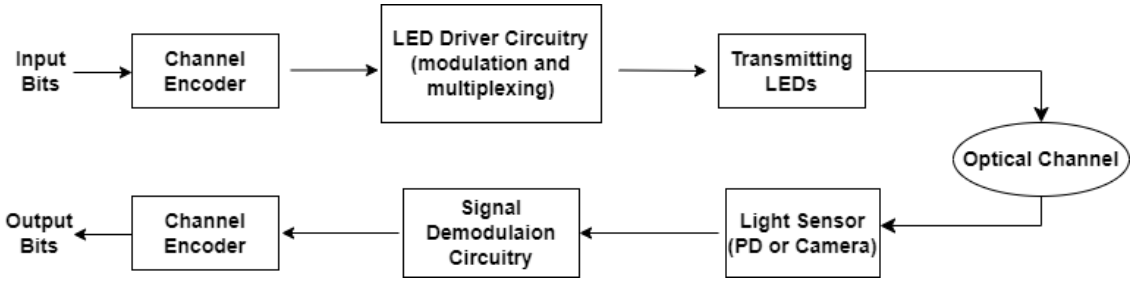
Các mô hình mạng được hỗ trợ từ lớp MAC bao gồm point-to-point (điểm – điểm), broadcast (một điểm – nhiều điểm) và star (mạng hình sao) [3]. Với cấu trúc mạng điểm – điểm, mối liên kết chỉ có hai đối tác tham gia, các đường truyền riêng được thiết lập để xây dựng một mạng truyền thông. Với cấu trúc mạng một điểm – nhiều điểm, trong một mối liên kết có nhiều đối tác tham gia, tuy nhiên chỉ một đối tác cố định duy nhất có khả năng phát trong khi nhiều đối tác còn lại thu nhận thông tin cùng một lúc. Với cấu trúc mạng hình sao, đây là một cấu trúc mạng có một trạm trung tâm quan trọng hơn tất cả các nút khác, nút này sẽ điều khiển hoạt động truyền thông của toàn mạng. Có thể nói, đây là kết hợp giữa cấu trục mạng điểm – điểm và một điểm – nhiều điểm. Hình 1. 4 thể hiện ba cấu trúc mạng trên.



Hình 1. 4 Cấu trúc mạng trong lớp MAC

b. Lớp Physical

Lớp vật lý (Physical layer) cung cấp các thông số vật lý của thiết bị và quan hệ giữa thiết bị và phương tiện truyền thông. Hình 1. 5 cho thấy sơ đồ khối của việc triển khai tầng vật lý tổng quát của hệ thống VLC (Visible Light Communication).



Hình 1. 5 Sơ đồ khối tầng vật lý hệ thống VLC

Đầu tiên, dữ liệu bits đầu vào được đưa qua kênh mã hóa (tùy chọn). Sau đó, quá trình điều chế (chẳng hạn như ON-OFF Keying, PPM và PWM) được thực hiện và dữ liệu được đưa vào đèn LED để truyền qua kênh quang (optical channel) [4].

Trong chuẩn IEEE 802.15.7, có ba loại lớp vật lý được cung cấp. Phạm vi hoạt động của lớp PHY I, PHY II, và PHY III lần lượt là 11.67-266.6 Kbps, 1.25-96 Mbps và 12-96 Mbps. Các hệ thống mã hóa kênh khác nhau, tốc độ quang và tốc độ dữ liệu được hỗ trợ bởi 802.15.7 có liệt kê trong Bảng 1. 1, Bảng 1. 2 và Bảng 1. 3 [3].

PHY 1 operating mode specifications

Modulation	RLL code	Optical clock rate	FEC		Data rate (kbps)
			Outer code (RS)	Inner code (CC)	
OOK	Manchester	200 kHz	(15,7)	1/4	11.67
			(15,11)	1/3	24.44
			(15,11)	2/3	48.59
			(15,11)	None	13.3
			None	None	100
VPPM	4B6B	400 kHz	(15,2)	None	35,56
			(15,4)	None	71.11
			(15,7)	None	124.4
			None	None	266.6

Bảng 1. 1 Lớp PHY I

PHY II operating mode specifications				
Modulation	**RLL code**	**Optical clock rate (MHz)**	**FEC**	**Data rate (Mbps)**
VPPM	4B6B	3.75	RS(64,32)	1.25
			RS(160,128)	2
		7.50	RS(64,32)	2.5
			RS(160,128)	4
			None	5
OOK	8B10B	15	RS(64,32)	6
			RS(160,128)	9.60
		30	RS(64,32)	12
			RS(160,128)	19.20
		60	RS(64,32)	24
			RS(160,128)	38.4
		120	RS(64,32)	48
			RS(160,128)	76.8
			None	96

Bảng 1. 2 Lớp PHY II

PHY III operating mode specifications			
Modulation	**Optical clock ra (MHz)**	**FEC**	**Data rate (Mbps)**
4-CSK	12	RS(64,32)	12
8-CSK		RS(64,32)	18
4-CSK	24	RS(64,32)	24
8-CSK		RS(64,32)	36
16-CSK		RS(64,32)	48
8-CSK		None	72
16-CSK		None	96

Bảng 1. 3 Lớp PHY III

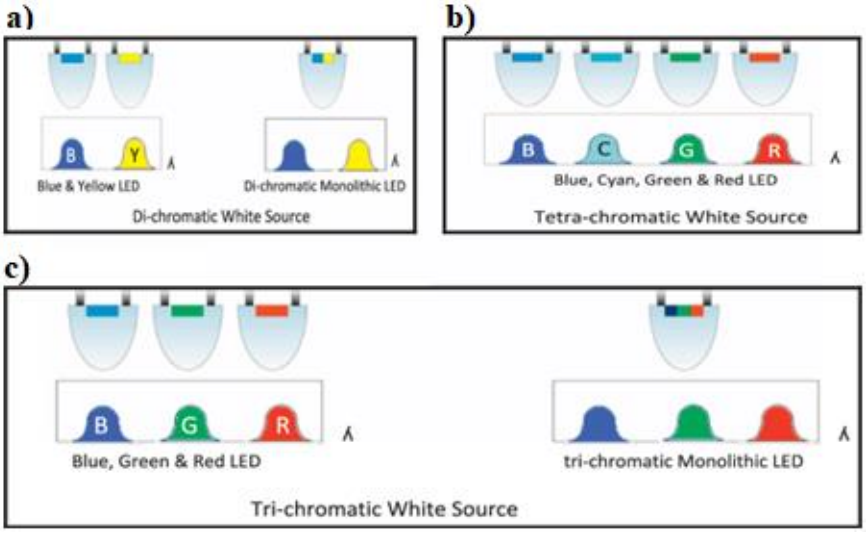
- Chế độ PHY I dành cho việc truyền thông ngoài trời – phương tiện giao thông. Trong PHY I, sử dụng mã Reed Solan (RS) để mã hóa kênh truyền thông
- Chế độ PHY II được thiết kế để sử dụng truyền thông trong nhà. PHY II hỗ trợ mã Run Length Limited (RLL) để giảm tác động nhấp nháy và cân bằng DC.
- Chế độ PHY III thì cũng được sử dụng trong môi trường trong nhà. Tương tự như PHY II, PHY III cũng hỗ trợ mã Run Length Limited (RLL).

c. Bộ phát (Tx)

Trong các nguồn sáng như đèn huỳnh quang, bóng đèn sợi đốt, đèn halogen và đèn LED hiện đang được sử dụng để chiếu sáng, chỉ có đèn LED là phù hợp làm bộ phát trong hệ thống VLC. Đèn LED đã vượt qua nguồn sáng bóng đèn truyền thống về độ tin cậy tốt hơn, tuổi thọ, hiệu suất sáng đều cao hơn, cung cấp giải màu rộng

hơn và có thể điều khiển được. Hiệu suất của đèn LED cao hơn 20 Lm/W so với đèn sợi đốt [5]. Ngoài ra, với khả năng bật và tắt ở tốc độ cao [6] của đèn LED được coi là nguồn sáng hoàn hảo cho VLC, đặc biệt là ứng dụng định vị [6].

Có các phương pháp khác nhau mà ánh sáng trắng có thể được tạo ra bởi các đèn LED. Các chế độ Tetra-chromatic, dichromatic và tri-chromatic được sử dụng để tạo ra ánh sáng trắng được thể hiện trong Hình 1. 6 [7].



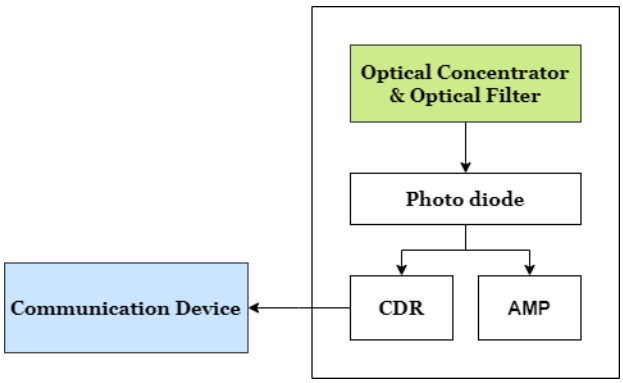
Hình 1. 6 Ba loại LED trắng

a) LED xanh vàng, b) LED RGB, c) Nhiều đèn LED RGB

Phương pháp phổ biến nhất được sử dụng để tạo ra ánh sáng trắng bằng đèn LED là ba màu (ví dụ là đỏ, xanh lá cây và xanh dương). Lợi ích của việc sử dụng đèn LED RGB để tạo ra ánh sáng trắng là băng thông cao và do đó, tốc độ truyền dữ liệu cao. Nhược điểm của đèn LED RGB là độ phức tạp cao và khó khăn trong việc điều chế.

d. Phần thu (Rx)

Bộ thu VLC điển hình bao gồm mạch khuếch đại, bộ lọc quang và bộ tập trung ánh sáng được hiển thị trong Hình 1. 7. Sự lan rộng của tia ánh sáng (beam divergence) trong đèn LED khi chiếu sáng vào các khu vực lớn dẫn đến suy giảm tín hiệu, do đó, bộ tập trung ánh sáng (optical concentrator) là thiết bị được sử dụng để bù đắp loại suy giảm này. Bằng cách sử dụng bộ tập trung ánh sáng, tia ánh sáng từ đèn LED được tập trung lại thành một tia hẹp hơn, giúp tăng cường cường độ ánh sáng và giảm suy giảm tín hiệu do lan rộng. Điều này giúp cải thiện hiệu suất truyền thông và đảm bảo tín hiệu nhận được có độ mạnh và độ ổn định tốt hơn. Trong bộ thu VLC như hình dưới, ánh sáng được phát hiện bằng cách sử dụng một photodiode và sau đó chuyển đổi thành dòng điện ánh sáng (photo current).



Hình 1. 7 Bộ thu VLC điển hình

Thông thường, các đi-ốt quang được sử dụng làm máy thu trong các hệ thống truyền thông ánh sáng nhìn thấy được. Tuy nhiên, đi-ốt quang cực kỳ nhạy cảm và bắt được các sóng nằm ngoài quang phổ của ánh sáng khả kiến, chẳng hạn như tia cực tím và hồng ngoại [8]. Chúng cũng dễ bão hòa, chẳng hạn như trong môi trường bên ngoài và tiếp xúc với ánh sáng mặt trời, và đi-ốt quang sẽ không nhận được dữ liệu do nhiễu cao. Vì lý do này, các thành phần khác có thể được sử dụng để thu ánh sáng. Một trong số đó là chính máy ảnh của điện thoại thông minh, cho phép bất kỳ điện thoại di động nào nhận dữ liệu được gửi bởi bộ phát VLC.

Một camera số gồm có ống kính, IS và các thành phần khác. Ống kính chụp ảnh là một thấu kính quang học được sử dụng để lấy nét ảnh của đối tượng vào IS. IS số là một mảng hai chiều các cảm biến quang, có chức năng phát hiện và truyền tín hiệu. IS theo công nghệ CMOS được sử dụng hầu hết trên các camera số chẳng hạn như trên điện thoại thông minh, thường được sử dụng để chụp mã vạch [9].

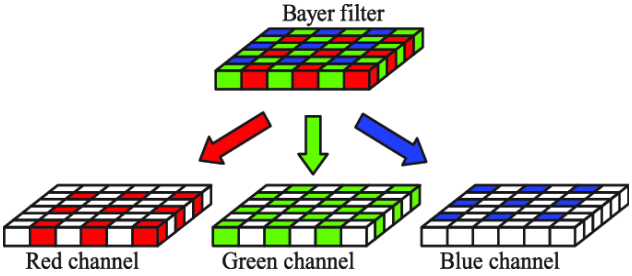
Giả sử tốc độ lấy mẫu đủ cao để tránh trường hợp răng cưa, thì đáp ứng xung và giá trị Fourier tương ứng của pixel trong CMOS được tính như sau:

$$h(t) = A(u(t) - u(t - T_{shutter}) \qquad PT\ 1.\ 1$$

$$H(f) = A\exp(-j\pi f\ T_{shutter}).\frac{2\sin\left(\pi f\ T_{shutter}\right)}{2\pi f} \qquad PT\ 1.\ 2$$

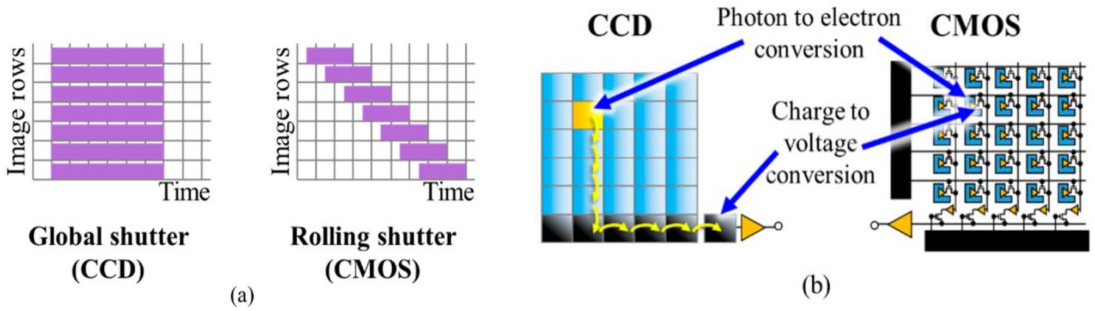
trong đó u(t) là xàm xung đơn vị, A là đáp ứng của PD và giá trị khuếch đại, $T_{shutter}$ là thời gian phơi sáng, $\sin(x)\ /x$ biểu diễn tần số đáp ứng của tín hiệu đầu vào trong khoảng thời gian $T_{shutter}$.

Để thu được thông tin về màu sắc, IS thường được thêm một bộ lọc đỏ, xanh lục hoặc xanh lam. Thông thường có thể sử dụng bộ lọc màu Bayer, là mẫu 2x2 như Hình 1. 8, để thu ảnh màu kỹ thuật số [10].



Hình 1. 8 Bộ lọc màu Bayer

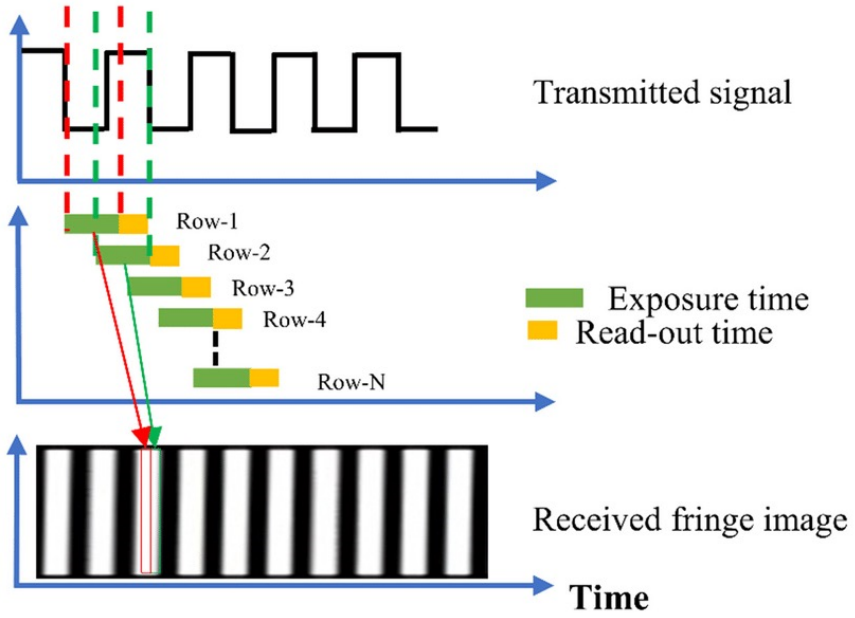
Trong camera, cơ chế màn trập sẽ xác định độ phơi sáng của pixel trong IS. Có hai loại IS chính bao gồm kiến trúc là CCD và CMOS. Cảm biến CCD có kích thước vật lý lớn hơn do sử dụng bộ chuyển đổi tương tự sang số (ADC) tương đối lơn hơn so với CMOS IS [[11], [12]]. Do đó, CCD thường không được triển khai trong các điện thoại di động, đặc biệt là điện thoại thông minh. So với CCD, máy ảnh CMOS cung cấp mức tiêu thụ điện năng thấp hơn, IS nhỏ hơn, khả năng đọc nhanh hơn, chi phí thấp hơn và khả năng lập trình cao hơn. Ngoài ra, do quét pixel dựa trên lược đồ địa chỉ X–Y trong kiến trúc CMOS, nên có thể truy cập trực tiếp vào phần IS mong muốn. Máy ảnh có thể có hai phương pháp phơi sáng khác nhau, tức là màn trập toàn cục (GS) và màn trập lăn (RS), như minh họa trong Hình 1. 9a. Trong IS dựa trên màn trập toàn cục, toàn bộ cảm biến được tiếp xúc với ánh sáng cùng một lúc, trong khi ở máy ảnh dựa trên màn trập lăn, mỗi hàng được tiếp xúc với ánh sáng tại một thời điểm, do đó giảm đáng kể độ sáng.



Hình 1. 9 Cấu trúc và cơ chế quét của cảm biến CCD và CMOS

Trong IS CCD dựa trên màn trập toàn cầu, thời gian chụp khung hình dài hơn đáng kể do để chuyển đổi điện tích thành tín hiệu điện áp [11]. Hình 1. 9b cho thấy cấu trúc của cảm biến CCD và CMOS [11]. Do chế tạo ADC nhỏ gọn hơn nhiều cho mỗi pixel, cảm biến CMOS ngày nay được ưa chuộng hơn so với các cảm biến CCD tương đối lớn hơn.

Hầu hết các cảm biến hình ảnh CMOS được nhúng trong điện thoại di động đều được vận hành bằng cách sử dụng màn trập lăn. Điều này có nghĩa là mỗi hàng điểm ảnh trong cảm biến CMOS được kích hoạt tuần tự và nó không chụp toàn bộ hình ảnh cùng một lúc (màn trập toàn cục) như hình minh họa trong Hình 1. 9. Mặc dù đây có thể là một hiệu ứng không mong muốn khi quay video đối tượng chuyển động nhưng nó có lợi cho hệ thống VLC vì tốc độ dữ liệu có thể nhanh hơn nhiều so với tốc độ khung hình (FPS). Do đó một hình ảnh có thể đại diện cho nhiều bit logic. Trong Hình 1. 10, ta có thể thấy chế độ làm việc của cảm biến hình ảnh CMOS yêu cầu thời gian phơi sáng và đọc dữ liệu được thực hiện bằng cách quét các pixel của từng dải sáng và tối theo từng hàng. Bằng cách chuyển đổi giữa các trạng thái ON và OFF của đèn LED trong khi truyền dữ liệu, các dải sáng và tối sẽ xuất hiện trong khung hình được chụp bởi cảm biến hình ảnh CMOS. Ngoài ra, giá trị Read – out time chính là thời gian cần thiết để đọc dữ liệu từ mỗi hàng pixel. Để đọc dữ liệu từ mỗi hàng pixel, tín hiệu điện áp tương ứng được chuyển thành tín hiệu số bằng một bộ chuyển đổi analog-to-digital converter (ADC). Thời gian cần thiết để đọc dữ liệu từ mỗi hàng pixel phụ thuộc vào tốc độ chuyển đổi của ADC và số lượng pixel trong cảm biến.



Hình 1. 10 Cảm biến CMOS quét đèn LED ở trạng thái ON và OFF

1.1.3 Đặc điểm của VLC

Hiện nay, các công nghệ truyền thông không dây sử dụng sóng vô tuyến như Wifi, Bluetooth, Zigbee… đã trở nên phổ biến nhưng vẫn còn những hạn chế nhất định về mức độ ảnh hưởng đến sức khỏe trẻ em, người già…Ngoài ra, phân bổ tần số trong dải sóng vô tuyến phụ thuộc vào quy định của từng quốc gia và được quản lý bởi các tổ chức viễn thông quốc tế, cùng với các nhu cầu dân dụng hoặc quân sự, chi phí cho hệ thống truyền thông bằng sóng vô tuyến trở nên rất đắt đỏ so với hệ thống VLC, nơi quang phổ ánh sáng miễn phí hoàn toàn. Với tính chất lan truyền, ánh sáng mang lại nhiều lợi thế về bảo mật so với sóng vô tuyến. Sóng vô tuyến có thể lan truyền xa hàng trăm mét qua các bức tường và vật cản rắn, gây nguy hiểm cho tính bảo mật. Ngược lại, ánh sáng lan truyền trong không gian phạm vi nhìn thấy, cung cấp môi trường bảo mật an toàn hơn. Bảng 1. 4 tổng hợp một số so sánh giữa truyền thông bằng ánh sáng nhìn thấy và truyền thông bằng sóng vô tuyến [13].

Property		VLC	RF
Bandwidth		Unlimited, 400nm~100nm	Regulatory, BW Limited
EMI		No	High
Line of Sight		Yes	No
Standard		IG-VLC	Yes
Hazard		No	Yes (H20 reaction to 2.4GHz)
Mobile to Mobile	Visibility (Security)	Yes	No
	Power Consumption	Relatively low	Medium
	Distance	Short	Medium
	Power Budget	Tight	Medium
Infra to Mobile	Security	Yes	No
	Intra	LED illumination	Access Point
	Mobility	Limited	Yes
	Coverage (Distance)	Short (~10m)	Wide (Short ~ Long Range)

Bảng 1. 4 So sánh đặc điểm giữa hệ thống VLC và truyền thông RF.

VLC có tiềm năng rất lớn vượt ra ngoài việc truyền dữ liệu không dây thông thường, đặc biệt là trong các môi trường trong nhà, nơi ánh sáng có thể được kiểm soát và ít bị ảnh hưởng bởi các yếu tố bên ngoài. Với việc sử dụng đèn LED ngày càng phổ biến và số lượng điện thoại di động trên toàn thế giới tăng theo cấp số nhân, nhu cầu sử dụng đèn LED được dự báo sẽ tiếp tục tăng trong thời gian tới, như được thể hiện trong Hình 1. 11

Hình 1. 11 Nhu cầu sử dụng đèn LED đến năm 2030

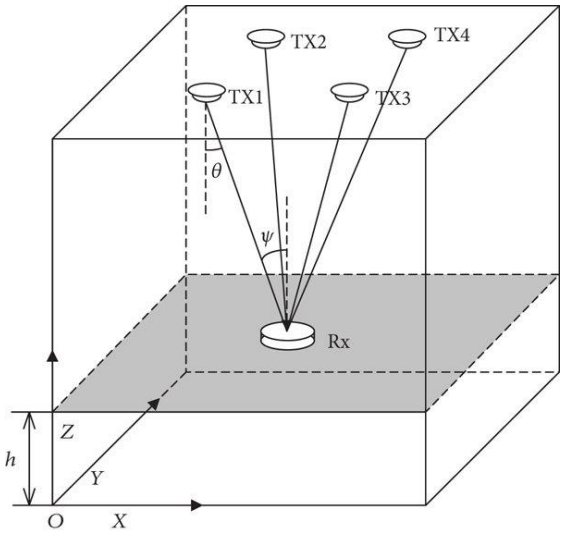
Bởi vì đèn LED có thể được lắp đặt ở nhiều vị trí khác nhau, việc tích hợp hệ thống VLC với các hệ thống chiếu sáng hiện có là dễ dàng trong khi vẫn đảm bảo tiết kiệm năng lượng và giảm phát thải khí nhà kính.

Vì vậy, VLC là một công nghệ tiềm năng cho các môi trường trong nhà cần ánh sáng và có thể phục vụ như một giải pháp thay thế khả thi cho việc truyền dữ liệu không dây bằng sóng vô tuyến.

1.1.4 Ứng dụng chung

a. Định vị trong nhà

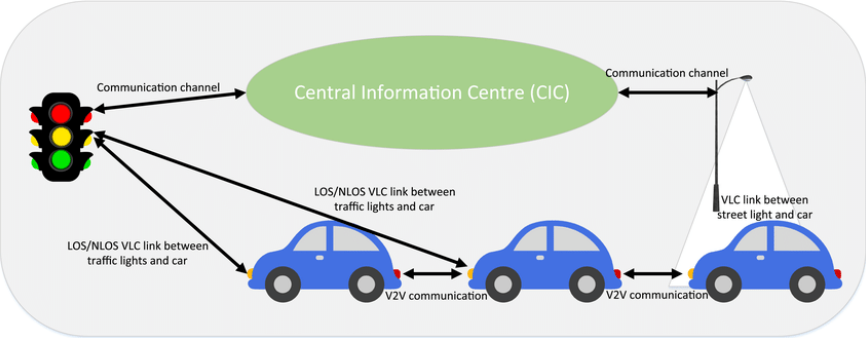
Một trong những ứng dụng tiềm năng cho VLC là định vị trong nhà, trung tâm thương mại sử dụng cơ sở hạ tầng chiếu sáng đèn LED. Đèn LED được cấp mã nhận dạng (ID) duy nhất, và thiết bị thông minh có tích hợp máy ảnh (ví dụ là điện thoại thông minh) để xác định vị trí người và vật trong không gian kín. Hình 1. 12 Mô tả cách bố trí của đèn LED trong một căn phòng. Ứng dụng này có thể được sử dụng trong tàu điện ngầm, bệnh viện và sân bay [1]. Khi điện thoại thông minh hướng thẳng đứng lên trên về phía đèn LED, máy ảnh có thể thu ánh sáng từ một hoặc nhiều đèn LED hoặc không có đèn LED nào cả. Từ đó, hệ thống sẽ truy vấn ID nhận được từ cơ sở dữ liệu để định vị vị trí đó rồi cho hướng dẫn đi đến vị trị cần tới.



Hình 1. 12 Ứng dụng định vị trong nhà bằng hệ thống VLC

b. Giao tiếp V2V và V2I

VLC có thể được sử dụng để liên lạc trên phương tiện giao thông do có đèn xe và cơ sở hạ tầng đèn giao thông hiện có. Các ứng dụng ưu tiên cao bao gồm cảnh báo va chạm phía trước, cảm biến trước va chạm, đèn phanh điện tử khẩn cấp, cảnh báo chuyển làn đường, hỗ trợ di chuyển biển báo dừng, hỗ trợ rẽ trái, cảnh báo vi phạm tín hiệu giao thông và cảnh báo tốc độ ở khúc cua [14]. Tất cả các ứng dụng có mức độ ưu tiên cao đều yêu cầu khả năng tiếp cận đáng tin cậy với độ trễ cực thấp. Với độ trễ cho phép cực thấp trong giao tiếp, hệ thống VLC cũng có tốc độ cao như Li-Fi. Dựa vào tín hiệu được trao đổi rất nhanh như vậy, hệ thống VLC có thể định vị và đo khoảng cách giữa các xe một cách đáng tin cậy trong phạm vi khoảng cách ngắn và đưa ra cảnh báo tới người lái.



Hình 1. 13 Phương tin giao thông giao tiếp bằng hệ thống VLC

c. Trong quảng cáo và giải trí

Với sự phát triển mạnh mẽ của các trung tâm giải trí và biển quảng cáo điện tử công cộng, đèn LED được sử dụng để tạo ra các biển quảng cáo lớn. Những bảng quảng cáo lớn như vậy có cường độ ánh sáng cao và được trang bị hàng trăm đèn LED, có thể được sử dụng để cung cấp kết nối mạng ngoài trời miễn phí và hoạt động như các điểm phát sóng cho người dùng, máy móc hoặc ứng dụng thành phố thông minh. Hình 1. 14 mô tả một bảng LED quảng cáo công cộng có kích thước lớn, tạo điều kiện thuận lợi cho việc phát triển VLC [15].

Hình 1. 14 VLC trong bảng quảng cáo sử dụng đèn LED

d. Li-Fi

Năm 2011, Harald Haas là người đầu tiên đặt ra thuật ngữ Light Fidelity (Li-Fi) [16]. Li-Fi là mạng hai chiều tốc độ cao được kết nối đầy đủ, có thể nhìn thấy hệ thống liên lạc không dây nhẹ và tương tự như Wi-Fi, sử dụng tần số vô tuyến để liên lạc [1]. Các tín hiệu Wi-Fi có vấn đề nhiễu với các tín hiệu RF khác như nhiễu với tín hiệu thiết bị dẫn đường của phi công trên máy bay. Vì vậy, ở những vùng nhạy cảm với bức xạ điện từ (chẳng hạn như máy bay) Li-Fi có thể là một giải pháp tốt hơn. Một Li-Fi cũng có thể hỗ trợ cho Internet vạn vật (IoT). Tốc độ lên tới 10Gbits/s thu được bằng Li-Fi, gấp 250 lần so với tốc độ của băng thông rộng siêu nhanh [1].

Hình 1. 15 Ứng dụng hệ thống VLC cho Li-Fi

1.2 Kỹ thuật điều chế cho hệ thống VLC

Điều chế trong VLC khác với điều chế trong truyền dữ liệu không dây RF do tính năng không mã hóa của thông tin về pha và biên độ của tín hiệu ánh sáng [1]. Do đó, rõ ràng là chúng ta không thể sử dụng điều chế biên độ và pha trong trường hợp VLC. Điều chế trong VLC đạt được bằng cách sử dụng các biến thể về cường độ ánh sáng tương ứng với thông tin trong tín hiệu truyền đi.

1.2.1 Các yếu tố ảnh hưởng đến việc điều chế trong VLC

Hai yếu tố được xem xét trong thiết kế sơ đồ điều chế cho VLC bao gồm: (1) làm mờ và (2) nhấp nháy

(1) Các hoạt động khác nhau đòi hỏi độ sáng khác nhau, chẳng hạn như độ sáng từ 30 đến 100 (lux) là cần thiết cho các hoạt động thị giác bình thường ở các địa điểm công cộng [1]. Có một mối quan hệ phi tuyến tính giữa ánh sáng đo được và ánh sáng cảm nhận được và mối quan hệ của chúng được cho bởi:

$$Perceived\ Light(\%) = 100 \times \sqrt{\frac{Measured\ Light(\%)}{100}} \qquad PT\ 1.3$$

(2) Những thay đổi về độ sáng của ánh sáng được điều chế nên được thực hiện sao cho không gây ra sự thay đổi đáng kể khiến cho mắt có thể cảm nhận được. Theo tiêu chuẩn IEEE 802.15.7, các thay đổi này nên được thực hiện với tốc độ nhanh hơn 200Hz để tránh tác động có hại về mắt.

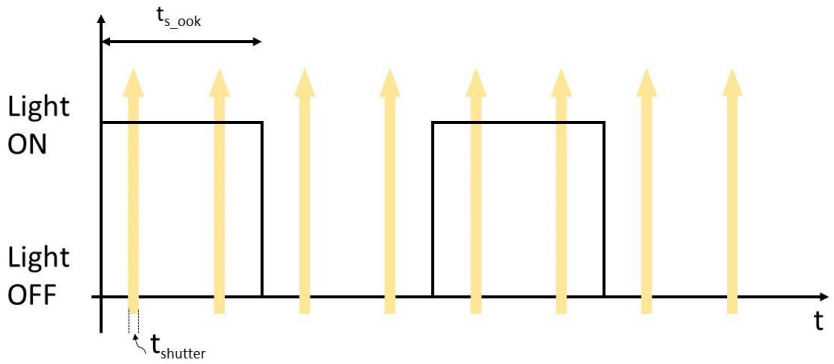
1.2.2 Kỹ thuật điều chế On-Off Keying (OOK)

a. Định nghĩa

Trong OOK, đèn LED được bật và tắt theo các bit trong chuỗi dữ liệu dựa vào cơ chế hoạt động màn trập lăn trong cảm biến ảnh để thực hiện. Ví dụ, số 1 được biểu thị bởi trạng thái bật và số 0 được biểu thị bởi trạng thái tắt. Trong OOK, đèn LED không được tắt hoàn toàn ở trạng thái tắt, nhưng mức độ cường độ sáng sẽ giảm đi. Lợi thế chính của việc sử dụng OOK là thiết kế và triển khai dễ dàng. Trong các nghiên cứu trước đây, VLC sử dụng kỹ thuật On-Off Keying được thực hiện bằng cách sử dụng đèn LED trắng (kết hợp giữa bộ phát xanh và phốt-pho màu vàng). Tuy nhiên, điều này gặp hạn chế về băng thông do thời gian phản hồi chậm của phốt-pho màu vàng [1]. Tốc độ dữ liệu 10Mbps đã được thử nghiệm trong

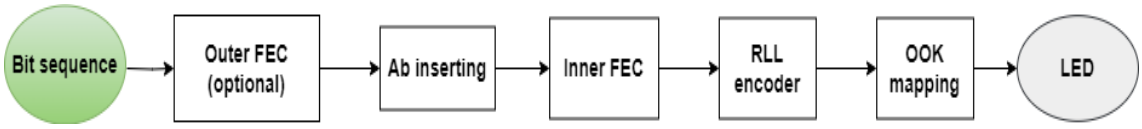
bằng NRZ (Non-Return-to-Zero) OOK với đèn LED trắng. Kết hợp giữa cân bằng tín hiệu và bộ lọc xanh đã được thực hiện để tăng tốc độ dữ liệu lên tới 125 Mbps và 100 Mbps tương ứng [1].

Để đảm bảo không có hiện tượng ánh sáng nhấp nháy, tần số của tín hiệu phải cao hơn mức tần số nháy (CFF). Do đó, tốc độ khung hình (tốc độ lấy mẫu) của camera phải ít nhất gấp đôi tần số từ Tx. Hình 1. 16 biểu diễn sơ đồ lấy mẫu của camera OOK, t_{s_ook} được lấy mẫu gấp hai lần khoảng thời gian lấy mẫu của màn trập $t_{shutter}$.



Hình 1. 16 Giải mã tín hiệu OOk vơi tần số lấy mẫu gấp đôi tần số TX

b. Hệ thống OOK

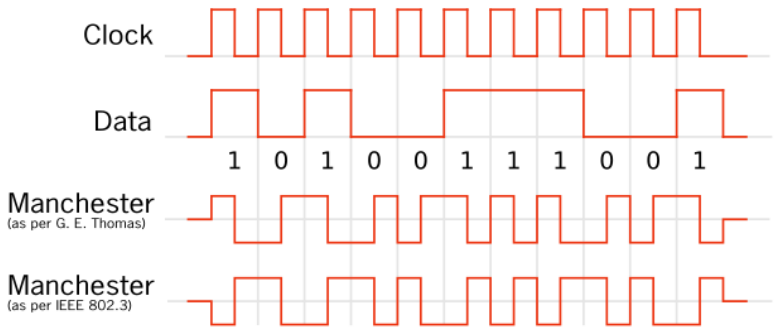


Hình 1. 17 Kiến trúc tham chiếu của OOK

Forward Error Correction (FEC): cần cho bất kỳ hệ thống truyền thông nào vì định dạng frame truyền của OOK bao gồm các gói con nên cần kết hợp việc sử dụng inner FEC và outer FEC để bảo vệ dữ liệu tránh bị lỗi. Inner FEC bảo vệ load ở trong gói con, trong khi outer FEC bảo vệ toàn bộ gói dữ liệu

Bit không đồng bộ (Ab): được tùy chỉnh bằng M-ary-FSK để xử lý sự thay đổi tốc độ khung hình của máy ảnh. Bất kỳ máy ảnh nào được tích hợp trong thiết bị cá nhân có hệ điều hành đều có vấn đề thay đổi tốc độ khung hình. Ở đây, Ab đóng vai trò là chuỗi số tối thiểu của gói PHY con để hỗ trợ quá trình lấy mẫu xuống phần thu khi độ lệch cao về khoảng thời gian lấy mẫu.

Mã RLL (Run-length limited): vì bản thân OOK không có sự cân bằng DC nên mã hóa này được sử dụng để đạt được cân bằng DC. Mã hóa Manchester được đề xuất trong điều chế OOK.



Hình 1. 18 Mã hóa Manchester trong điều chế OOK

Việc sử dụng mã Manchester trong điều chế OOK có nhiệm vụ chính là cân bằng số lượng bit 0 và bit 1. Do dữ liệu được mã hóa theo độ rộng xung nên thông tin được gửi sẽ ảnh hưởng đến mức độ mờ nếu không được sửa.

c. Phương thức giải điều chế OOK

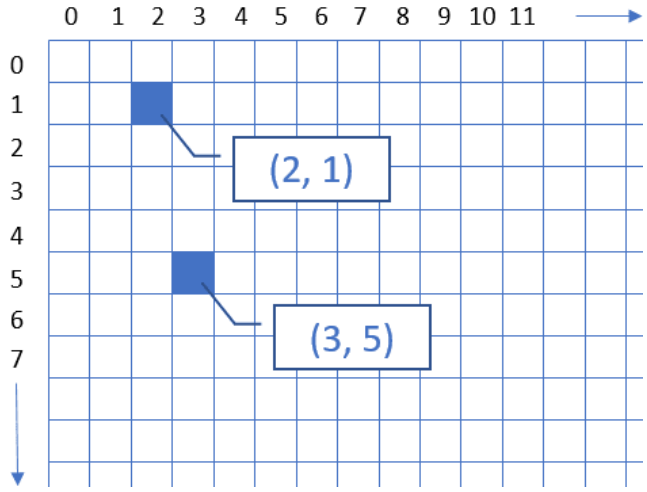
Để giải điều chế toàn bộ gói dữ liệu con DS, khoảng cách từ máy ảnh đến bộ phát LED phải đủ gần. Khoảng cách tối đa đạt được là khoảng cách mà máy ảnh nhận được lượng dữ liệu bằng với lượng của gói con. Khi đó, máy ảnh phát hiện ký hiện mở đầu và giải điều chế lượng dữ liệu đủ cho một gói con. Tuy nhiên, sẽ có trường hợp phần trước và sau tính tử vị trí của phần mở đầu sẽ không thuộc về một gói con. Đây có thể là vấn đề nhận không đủ dữ liệu, thậm chí là ít dữ liệu xảy ra ở khoảng cách ngắn, hoặc gói con có rất nhiều dữ liệu.

1.3 Thị giác máy và phần mềm

1.3.1 Tổng quan về ảnh kỹ thuật số

a. Điểm ảnh (Picture Element)

Điểm ảnh (Pixel) là một đơn vị nhỏ nhất của ảnh số với mỗi toạ độ điểm ảnh *(x, y)* có độ xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh đó được chọn thích hợp sao cho mắt người cảm nhận sự liên tục về không gian và mức xám (hoặc màu) của ảnh số gần như ảnh thật. Mỗi phần tử trong ma trận được gọi là một phần tử ảnh.



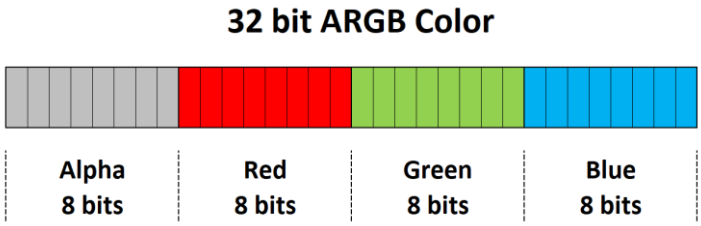
Hình 1. 19 Tọa độ pixel trong một ảnh

Trong Hình 1. 19, có thể thấy ví dụ về hai pixel lần lượt có vị trí tọa độ (2,1) và (3.5). Với từng vị trị tọa độ của pixel, nó sẽ có giá trị tương ứng để biểu thị độ sáng của pixel. Độ sáng (hay giá trị) của một pixel phụ thuộc vào loại hình ảnh ta sử dụng. Giá trị tối đa cho các pixel hình ảnh thường liên quan đến đặc tính của máy ảnh gọi là độ sâu bit (bit depth). Với độ sâu bit là k, có nhiều nhất là 2^k mức độ sáng có thể xác định. Ví dụ nếu độ sâu bit là 8 bit, một pixel có thể có 256 giá trị (2^8) trong phạm vi từ 0 đến 255. Nếu điểm ảnh có giá trị bằng 0 thì nó có màu đen nhất, còn giá trị bằng 255 tương ứng với màu trắng nhất.

Trong ảnh kỹ thuật số, có 3 loại hình ảnh mà có thể xử lý: ảnh nhị phân, ảnh xám và ảnh màu. Mỗi loại ảnh đều có đặc điểm riêng và cách xử lý ảnh khác nhau, cụ thể sẽ được trình bày ở những nội dung tiếp theo.

b. Ảnh màu

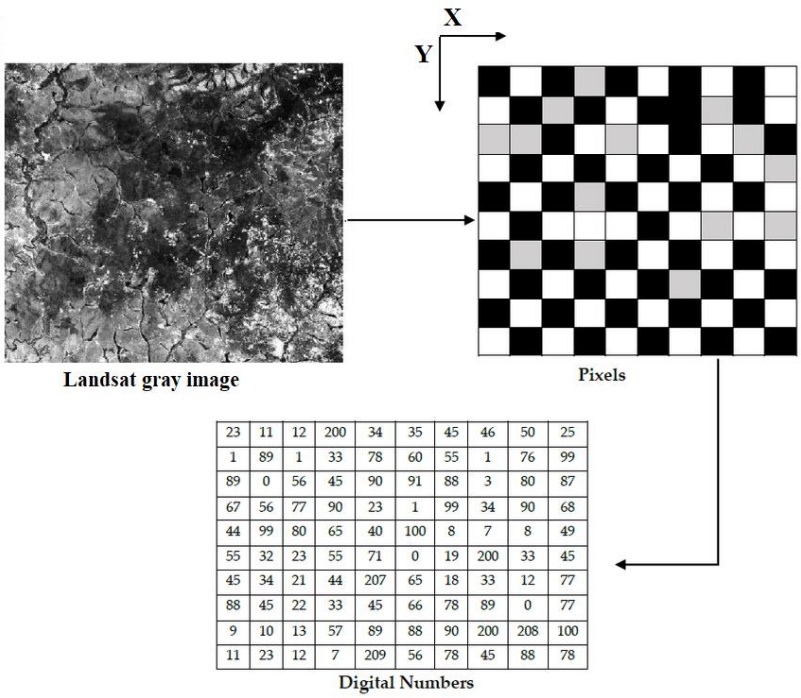
Ảnh màu lấy từ máy ảnh kỹ thuật số thường được mô tả bằng ba giá trị màu: R (Red – đỏ), G (Green – xanh lá cây) và B (Blue – xanh lam). Ba giá trị màu đại diện cho một pixel ảnh mô tả màu sắc và độ sáng của pixel và trong ảnh kỹ thuật số thường sẽ kết hợp các giá trị R, G và B để tạo ra một giá trị của một pixel. Ảnh màu thường được gọi là ảnh 24 bits hoặc 32 bits. Hình 1. 20 cho thấy khái niệm về ảnh màu 32 bits. Trong đó có bốn giá trị 8 bits gồm bốn thành phần R, G, B và một kênh Alpha để tạo độ sáng tối, đậm nhạt.



Hình 1. 20 Ảnh màu 32 bits

c. Ảnh xám (gray image)

Ảnh xám hay còn gọi là ảnh đơn sắc (monochromatic). Một pixel trong ảnh xám nếu được mã hóa bằng 8-bit thì số các mức xám có thể biểu diễn được là 256. Mỗi mức xám (hay giá trị) được biểu diễn dưới dạng là một số nguyên trong khoảng từ 0 đến 255. Nếu giá trị pixel là 0 thì vị trí đó tối nhất, còn 255 là sáng nhất. Ở Hình 1. 21, một phần ảnh xám được phóng đại để có thể thấy rõ từng pixel. Tương ứng từng vị trí của pixel là giá trị của nó.

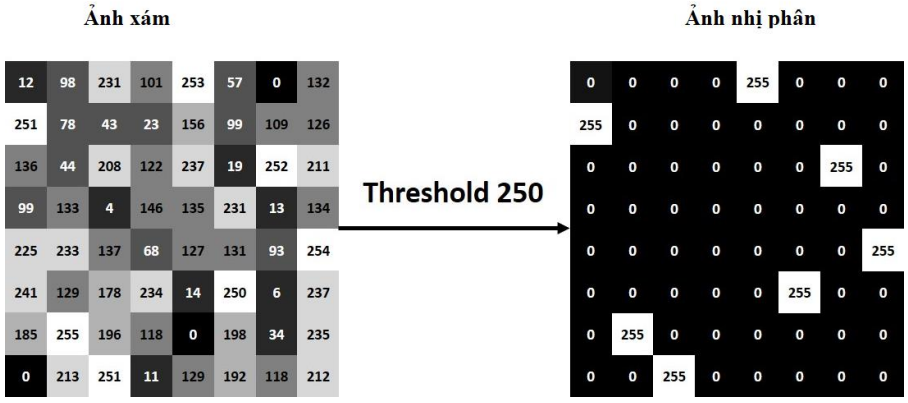


23	11	12	200	34	35	45	46	50	25
1	89	1	33	78	60	55	1	76	99
89	0	56	45	90	91	88	3	80	87
67	56	77	90	23	1	99	34	90	68
44	99	80	65	40	100	8	7	8	49
55	32	23	55	71	0	19	200	33	45
45	34	21	44	207	65	18	33	12	77
88	45	22	33	45	66	78	89	0	77
9	10	13	57	89	88	90	200	208	100
11	23	12	7	209	56	78	45	88	78

Hình 1. 21 Ảnh xám 8 bits

d. Ảnh nhị phân (binary image)

Ảnh nhị phân (bên phải Hình 1. 22) là mỗi điểm ảnh được biểu diễn bởi giá trị là 0 (đen) và 255 (trắng) tương ứng với 0 và 1. Vì chỉ có hai giá trị, nên nó thường đường gọi là ảnh 1-bit (độ sâu là bit là 1).



Hình 1. 22 Ảnh nhị phân chuyển từ ảnh xám

Ảnh nhị phân là định dạng ảnh được sử dụng phổ biến nhất để phát hiện đối tượng, vị trí và kích thước. Để có được ảnh nhị phân, thì ta thường sử dụng ảnh xám để tạo ảnh nhị phân. Quá trình biến đổi từ một ảnh xám sang ảnh nhị phân (Hình 1. 23) thì sẽ cần đến một giá trị ngưỡng (threshold), cụ thể như sau:

- Gọi giá trị cường độ sáng tại một điểm là I (x, y)
- INP (x, y) là cường độ sáng của điểm ảnh trên ảnh nhị phân
- Điều kiện: 0 < x < width và 0 < y < height (width: chiều rộng của ảnh, height: chiều dài của ảnh)

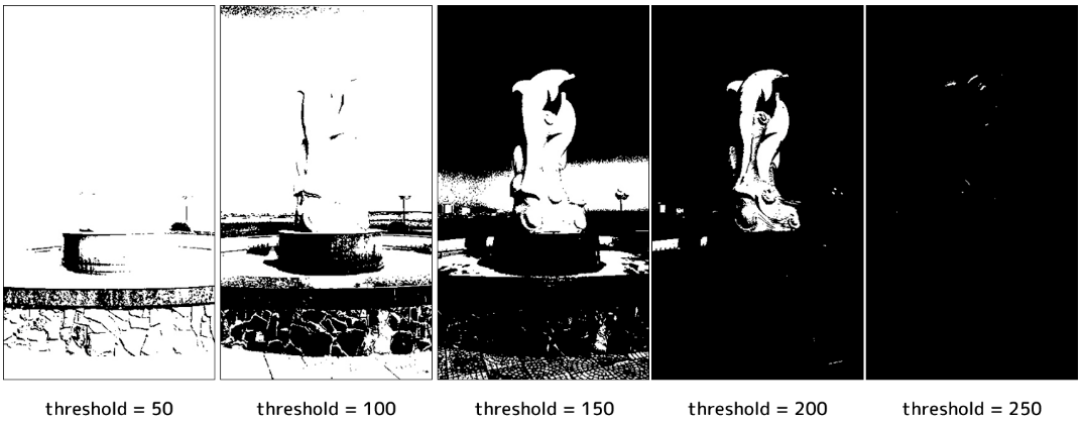
Sau đó, ta so sánh giá trị cường độ sáng của điểm ảnh với một ngưỡng nhị phân T:

- Nếu $I(x, y) < T$ thì $INP(x, y) = 0$ (0)

- Nếu $I(x,y) < T$ thì $INP(x,y) = 255$ (1)

Giá trị T ta có thể chọn từ 0 đến 255, nhưng thông thường sẽ chọn một giá trị là 255 tức là giá trị trung bình của max (255) và min (0) của cường độ sáng (Intensity) của điểm ảnh.

Hình 1. 23 Quá trình chuyển sang ảnh nhị phân



Hình 1. 24 Ảnh nhị phân với từng giá trị ngưỡng khác nhau

Có thể thấy rằng, với mỗi giá trị T (threshold) thì sẽ cho ra một ảnh nhị phân khác nhau được thể hiện trong Hình 1. 24. Do đó, để tối ưu trong việc tiền xử lý ảnh, lựa chọn giá trị ngưỡng cũng rất quan trọng. Một số phương pháp, thuật toán giúp biến đổi một ảnh xám thành ảnh nhị phân hay còn gọi nhị phân hóa (Adaptive Threshold) tự động, tối ưu hơn thay vì lựa chọn trung bình giá trị max và min cường độ sáng của mỗi điểm ảnh. Sau khi ảnh xám được chuyển đổi thành ảnh nhị phân, ta có thể thực hiện nhiều chức năng xử lý ảnh khác nhau. Ví dụ, có thể sử dụng chức năng phân tích phần tử (particle analynis), tìm contours để thu được kích thước, diện tích, và trọng tâm của vật thể.

1.3.2 Thị giác máy trong VLC

a. Định nghĩa

Thị giác máy trên điện thoại di động thông minh là việc sử dụng các kỹ thuật và công nghệ thị giác máy để phát triển các phần mềm trên chúng. Thị giác máy có

thể được sử dụng để nhận dạng khuôn mặt, phát hiện đối tượng, nhận dạng văn bản, phát hiện chuyển động và các tác vụ thị giác máy khác. Hiện nay, trên thị trường có nhiều hệ điều hành được sử dụng trên các điện thoại di động thông minh, bao gồm Android, IOS, Windows Phone và các hệ điều hành khác. Mỗi hệ điều hành có những đặc điểm và tính năng riêng, tuy nhiên, các công nghệ và thư viện thị giác máy được sử dụng phổ biến trên tất cả các nền tảng.

b. Ưu điểm

Ưu điểm của thị giác máy trên các điện thoại di động thông minh:

- Tiện lợi và di động: thị giác máy trên các điện thoại di động thông minh cho phép người dùng sử dụng các phần mềm thị giác máy bất cứ khi nào và ở bất cứ đâu mà không cần phải sử dụng máy tính hoặc thiết bị đặc biệt.
- Tính năng tiên tiến: các công nghệ và thư viện thị giác máy trên các điện thoại di động thông minh được phát triển liên tục và cung cấp các tính năng tiên tiến như nhận dạng khuôn mặt, phát hiện đối tượng và nhận dạng văn bản.
- Tính tương thích: các phần mềm thị giác máy trên các điện thoại di động thông minh được phát triển để hoạt động trên nhiều nền tảng và hệ điều hành khác nhau, giúp người dùng có thể sử dụng trên nhiều thiết bị khác nhau.

Với những ưu điểm của thị giác máy trên điện thoại di động thông minh đã trình bày, dễ thấy ứng dụng về việc định vị thông qua ánh sáng nhìn thấy được dựa trên thị giác máy có thể trở thành một công nghệ tiết kiệm chi phí để định vị chính xác trong các không gian trong nhà. Ý tưởng của phương pháp này là sử dụng camera có độ phân giải cao trên các thiết bị di động như máy tính bảng và điện thoại di động, và sau đó sử dụng quá trình xử lý ảnh để phân tích và định vị vị trí của thiết bị trong hệ thống VLC. Điều này có thể giúp giảm chi phí và cải thiện độ chính xác của hệ thống định vị trong các môi trường trong nhà.

1.3.3 Ngôn ngữ, thư viện và nền tảng hỗ trợ

Để hỗ trợ cho công nghệ thị giác máy trên điện thoại di động thông minh, các nhà phát triển phần mềm cũng cung cấp các ngôn ngữ lập trình và thư viện xử lý ảnh phổ biến trên điện thoại thông minh bao gồm:

- Java/Kotlin: Java là một trong những ngôn ngữ lập trình phổ biến nhất trên hệ điều hành Android và Kotlin là ngôn ngữ lập trình mới hỗ trợ lập trình ứng dụng Android. Java/Kotlin hỗ trợ các thư viện xử lý ảnh và đồ họa như OpenCV, Android Bitmap, OpenGL ES, Android Camera API, và Android Media API.
- Swift: Swift là ngôn ngữ lập trình chính trên hệ điều hành IOS và được sử dụng trong việc phát triển các ứng dụng trên iPhone, iPad và Mac. Swift hỗ trợ các thư viện xử lý ảnh và đồ họa như Core Image, Core Graphics, Metal, và AVFoundation.

- Python: Python là một trong những ngôn ngữ lập trình phổ biến trong lĩnh vực xử lý ảnh và đồ họa. Python hỗ trợ các thư viện xử lý ảnh như OpenCV, Scikit-image, Numpy. Các thư viện này cung cấp các chức năng xử lý ảnh đầy đủ và dễ sử dụng. Python có tính tương thích cao với các hệ thống khác nhau và hỗ trợ nhiều nền tảng khác nhau như Windows, macOS, và Linux. Điều này giúp cho các nhà phát triển có thể phát triển các ứng dụng xử lý ảnh trên nhiều hệ thống khác nhau một cách dễ dàng. Ngoài ra, Python có các thư viện học sâu như TensorFlow và PyTorch, giúp cho các nhà phát triển có thể phát triển các ứng dụng xử lý ảnh thông minh, như nhận dạng đối tượng và nhận dạng khuôn mặt.
- OpenCV: OpenCV là một thư viện xử lý ảnh mã nguồn mở được sử dụng rộng rãi trong nhiều ứng dụng xử lý ảnh trên điện thoại thông minh. OpenCV hỗ trợ nhiều tính năng xử lý ảnh như phát hiện đối tượng, nhận dạng khuôn mặt, đọc và ghi ảnh, và xử lý video.
- TensorFlow và PyTorch: TensorFlow và PyTorch là các thư viện học máy phổ biến được sử dụng trong lĩnh vực xử lý ảnh trên điện thoại thông minh. Cả hai thư viện hỗ trợ xử lý ảnh và đồ họa, và được sử dụng rộng rãi trong việc phát triển các ứng dụng nhận dạng đối tượng và nhận dạng khuôn mặt.
- C/C++: C/C++ là các ngôn ngữ lập trình được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và đồ họa. C/C++ hỗ trợ các thư viện xử lý ảnh như OpenCV và thư viện đồ họa OpenGL ES.

Trong đề tài này, em lựa chọn hai ngôn ngữ lập trình là **Python** và **Kotlin** cùng với thư viện chính là **OpenCV** để xây dựng ứng dụng di động có sử dụng công nghệ thị giác máy (xử lý ảnh) được lập trình trên **Android Studio IDE**. Ngôn ngữ Python sẽ có nhiệm vụ chính là thực hiện tác vụ xử lý ảnh với những ưu điểm vượt trội đã trình bày trước đó. Lựa chọn ngôn ngữ Kotlin là do được hỗ trợ chính thức trong Android Studio và Android SDK cùng với sự đa năng để phát triển ứng dụng đa nền tảng. Python không phải là một ngôn ngữ lập trình chính thức được hỗ trợ trong Android Studio. Tuy nhiên, có một số cách để tích hợp Python vào các ứng dụng Android được phát triển bằng Android Studio như sử dụng Chaquopy.

Android Studio là một IDE (Integrated Development Environment - Môi trường Phát triển Tích hợp) được Google phát triển, xây dựng dựa trên nền tảng của IntelliJ IDEA, được sử dụng để phát triển các ứng dụng Android. Nó cung cấp cho các nhà phát triển một loạt các tính năng và công cụ để hỗ trợ quá trình phát triển ứng dụng Android. Các tính năng nổi bật của Android Studio bao gồm một trình biên tập mã tích hợp, cho phép các nhà phát triển viết và chỉnh sửa mã nguồn Java và Kotlin, cùng với một trình tạo giao diện người dùng (UI Designer) giúp cho các nhà phát triển có thể thiết kế giao diện người dùng cho ứng dụng của mình dễ dàng hơn. Ngoài ra, Android Studio còn cung cấp một emulator Android tích hợp, giúp cho các nhà phát triển có thể kiểm tra và thử nghiệm ứng dụng của mình trên nhiều thiết bị khác nhau.

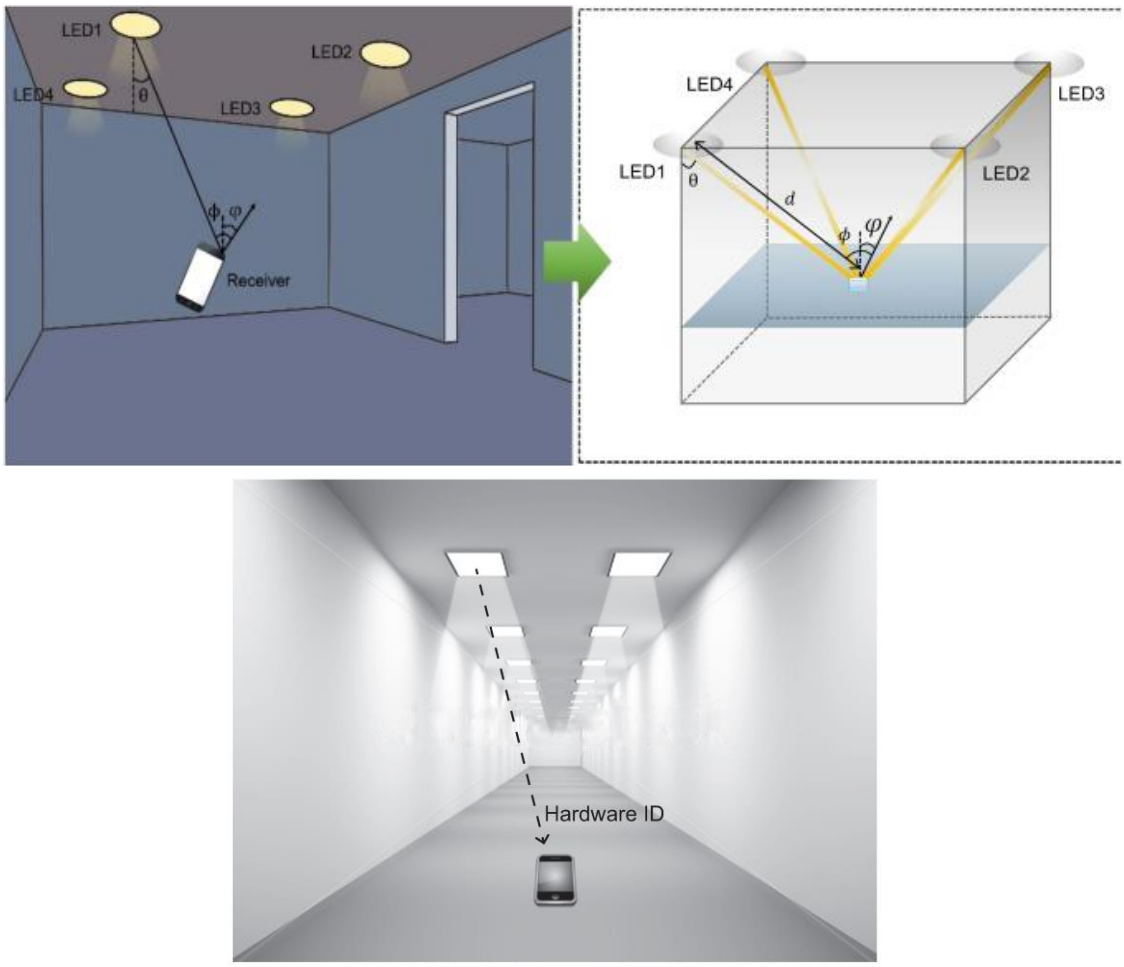
Chaquopy là một thư viện giúp tích hợp Python vào ứng dụng Android. Chaquopy cho phép các nhà phát triển tạo các module Python và tích hợp chúng vào ứng

dụng Android bằng cách sử dụng plugin được cung cấp cho Android Studio. Plugin này cho phép các nhà phát triển tạo một thư mục chứa các tệp tin Python và tệp tin Gradle để cấu hình và quản lý các phụ thuộc của module Python. Sau đó, các module Python này có thể được sử dụng trong mã Java hoặc Kotlin của ứng dụng Android, giúp cho các nhà phát triển có thể tích hợp các tính năng của Python vào ứng dụng của mình. Ngoài ra, Chaquopy cũng hỗ trợ các thư viện Python phổ biến như OpenCV, NumPy và SciPy.

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG

2.1 Tổng quan mô hình hệ thống

Mô hình này được thiết kế để áp dụng cho các tòa nhà nhiều tầng trong đó mỗi phòng được trang bị một số lượng đèn LED đơn hoặc theo cụm để phục vụ các mục đích chiếu sáng khác nhau. Hành lang có thể được trang bị bằng các dãy đèn đơn để tăng cường ánh sáng.



Hình 2. 1 Mô hình hệ thống VLC trong ứng dụng định vị trong nhà

Trong hệ thống này, các đèn LED không chỉ được sử dụng để chiếu sáng mà còn để phát tín hiệu ID chứa thông tin về địa chỉ phòng, vị trí đang đứng tại hành lang và vị trí của chúng. Bằng cách so sánh dữ liệu được thu thập từ máy ảnh trên điện thoại di động (hoặc máy tính bảng) của người dùng với dữ liệu phát ra từ các đèn cụ thể, hệ thống có thể tính toán chính xác vị trí góc lệch và hướng di chuyển của người dùng. Điều này được thể hiện rõ ràng trong Hình 2. 1.

Để mô phỏng hệ thống ở từng phòng, ta đặt tên là “hệ thống con”, em sẽ có hai mô hình chiếu sáng sử dụng điện thoại để nhận và giải dữ liệu gồm: chiếu sáng cụm đèn LED tròn và chiếu sáng đèn LED panel như Hình 2. 2, thực nghiệm tại phòng LAB.

a)

b)

Hình 2. 2 Mô phỏng thực nghiệm tại phòng LAB

2.2 Thiết kế hệ thống

2.2.1 Hệ thống Tx

a. Thông số kỹ thuật

- Đèn LED sử dụng:

Hình 2. 3 LED tròn

Hình 2. 4 LED panel

Tên thông số	Giá trị	
	LED panel	LED tròn
Loại và hãng LED	LED âm trần, hãng Kingled, màu đèn: trắng	
Công suất LED	LED panel: 48w	LED tròn: 12w
Quang thông	4320 Lm	1200 Lm
Nguồn	20 - 40 VDC	20 – 40 VDC
Góc chiếu	120 độ	120 độ
Kích thước	300 x 1200 x10 mm	đường kính đèn 10 cm kích thước đèn Ø 140*30mm
Số lượng LED	1	4
Xuất xứ	Việt Nam	

Bảng 2. 1 Thông số 2 loại LED sử dụng

- Vi điều khiển ESP32-WROVER

Module ESP32-WROVER là một vi điều khiển mã nguồn mở tích hợp chip ESP32-D0WDQ6 của Espressif Systems, được sử dụng trong các ứng dụng WiFi và Bluetooth Low Energy. Có hai lý do chính để lựa chọn sử dụng vi điều khiển này. Đầu tiên, nó là một trong những vi điều khiển phổ biến và được sử dụng rộng rãi trong lĩnh vực nghiên cứu tại Việt Nam. Thứ hai, việc lập trình cho chip này rất dễ dàng nhờ vào sự hỗ trợ của IDE và bộ thư viện đa dạng.

Vi điều khiển	ESP32-WROVER
Điện áp hoạt động	2.3 – 3.6V
Tần số hoạt động	40MHz
Tần số Wifi	2.4 – 2.5 GHz
PSRAM	8MB
SPI flash	4MB
ROM	448 KB
SRAM	520 KB
Dòng điện sử dụng	500mA

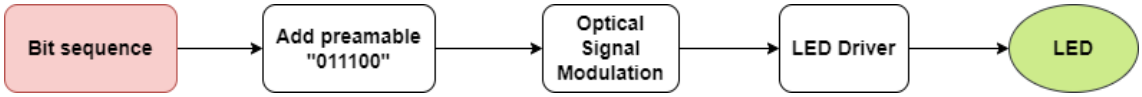
Bảng 2. 2 Thông số module ESP32 - WROVER

b. Cách truyền dữ liệu

Sơ đồ truyền dữ liệu của đèn LED được mô tả chi tiết trong Hình 2. 5. Ban đầu, một chuỗi 2 bits hoặc 4 bits chứa mã ID cần truyền sẽ được tạo ra. Sau khi áp dụng kỹ thuật mã hóa Manchester, độ dài của chuỗi sẽ tăng gấp đôi lên 4 bits (ứng với 2 bits gốc) hoặc 8 bits (ứng với 4 bits gốc). Trước khi truyền tín hiệu, thêm 6 bit preamable vào chuỗi để thu được chuỗi hoàn chỉnh có độ dài 10 bits hoặc 14 bits.

Module ESP 32 sẽ đóng vai trò làm driver cho đèn LED bằng cách tạo ra xung PWM dựa trên chuỗi bits đã được tạo ra trước đó. Tần số của xung này có thể điều chỉnh được thông qua ngắt timer/counter.

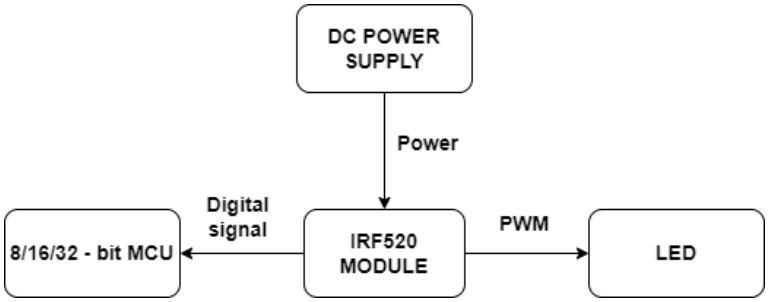
Đèn LED sẽ phát tín hiệu theo chuỗi bit đã được lập trình trước đó. Tần số của đèn LED được chọn là 1kHz để tránh khi nháy không tác động đến thị giác của người dùng.



Hình 2. 5 Sơ đồ điều chế Tx

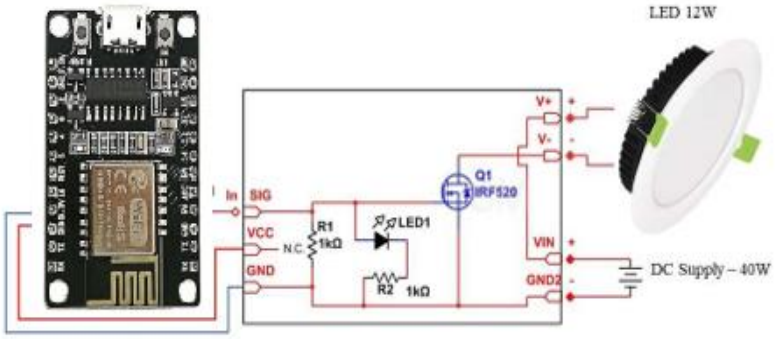
c. Sơ đồ phần cứng

Sơ đồ khối của Tx trong Hình 2. 6



Hình 2. 6 Sơ đồ phần cứng của Tx

Sơ đồ nguyên lí của Tx sẽ theo Hình 2. 7:



Hình 2. 7 Sơ đồ nguyên lý của Tx

d. Cấu trúc gói dữ liệu

Bảng 2. 3 đưa ra cấu trúc gói dữ liệu (frame truyền) của Tx, bao gồm 6 bits preambale và 4 bits hoặc 8 bits dữ liệu sau khi đã sử dụng mã hóa Manchester. Phần dữ liệu này sẽ chứa ID của địa chỉ phòng hoặc địa chỉ đang đứng trong hành lang.

Preamable	Payload (sử dụng mã hóa Manchester)
011100	2B ⟶ 4B
011100	4B ⟶ 8B

Bảng 2. 3 Cấu trúc gói dữ liệu

2.2.2 Hệ thống Rx

a. Các thông số ảnh hưởng đến Rx

- Tốc độ khung hình (Frame rate)

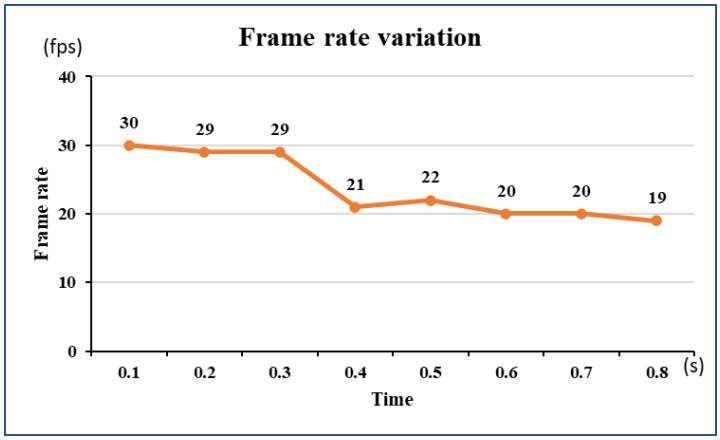
Tốc độ khung hình là số lượng ảnh thu được trong một giây, đơn vị là fps. Hầu hết các camera điện thoại di động thông minh hiện nay thông thường có tốc độ khung hình 24 fps, 30fps, 60fps và một vài dòng smartphone cao cấp hơn thì lên tới 120 hoặc 240 fps. Với tốc độ khung ảnh như vậy đủ để cho người dùng không cảm thấy bị giật cục khi ghi lại các hiệu ứng chuyển động. Tốc độ khung hình cũng phụ thuộc một phần vào tốc độ lưu trữ và bộ xử lý của điện thoại.

Tốc độ khung hình của camera đóng vai trò quan trọng trong VLC bởi nó ảnh hưởng đến tốc độ dữ liệu trong VLC. Về cơ bản, tốc độ dữ liệu VLC có thể được tính dựa trên tốc độ khung hình của camera như sau [17]:

$$R_{bps} = R_{fps} \times N_{bpf} \qquad PT\ 2.\ 1$$

trong đó R_{bps} là tốc độ bit (bit/giây), R_{fps} là tốc độ khung hình của camera, N_{bpf} là số bit trên mỗi khung hình.

Hình 2. 8 biểu diễn tốc độ khung hình thực tế của camera điện thoại di động. Tốc độ khung hình camera thực tế có thể được tính bằng cách đo khoảng thời gian giữa hai hình ảnh được chụp liên tiếp.



Hình 2. 8 Đo sự thay đổi tốc độ khung hình của điện thoại Google Pixel 4 khi trước và sau khi áp dụng quá trình xử lý ảnh

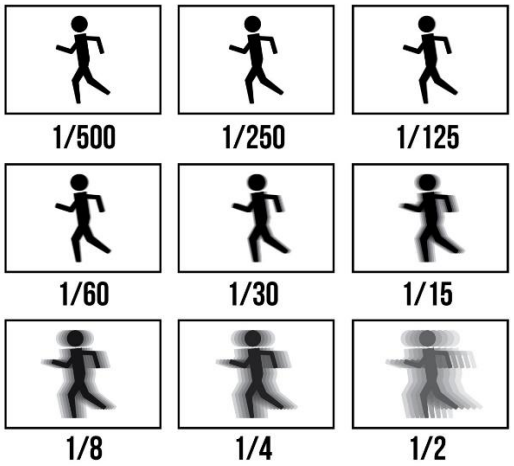
Trong quá trình đo, tốc độ khung hình ở phần thu có thể bị sụt giảm trong quá trình xử lý. Ví dụ ban đầu, tốc độ khung hình ở máy ảnh đang đạt 30 fps nghĩa là bình thường camera ghi lại 30 khung hình trong một giây khi chưa áp dụng thuật toán xử lý hình ảnh. Tuy nhiên, khi áp dụng tác vụ xử lý nặng để phát hiện và giải điều chế dữ liệu, tốc độ khung hình đôi khi giảm xuống còn 20 fps.

- **Tốc độ màn trập (Shutter speed)**

Máy ảnh hiện đại có thể chia làm hai loại: loại sử dụng màn trập cơ học và loại sử dụng màn trập điện tử. Đối với các loại máy điện thoại di động hiện nay, các nhà sản xuất đã thiết kế không có màn trập cơ học như máy ảnh truyền thống. Thay vào đó, điện thoại thông minh có màn trập máy ảnh điện tử, hoạt động bằng cách bật và tắt cảm biến trong thời gian phơi sáng. Một vài điện thoại có chế độ điều khiển được tốc độ màn trập với cách chỉnh thông số thời gian phơi sáng của camera như trong Hình 2. 9

Hình 2. 9 Điều chỉnh thời gian phơi sáng trong camera điện thoại

Tốc độ màn trập được tính bằng giây hoặc phần của một giây như 1/60, 1/125, 1/250. Thời gian phơi sáng càng lâu thì sẽ có càng nhiều ánh sáng lọt vào cảm biến, làm bức ảnh sáng hơn tuy nhiên sẽ gây ra hiệu ứng mờ chuyển động.



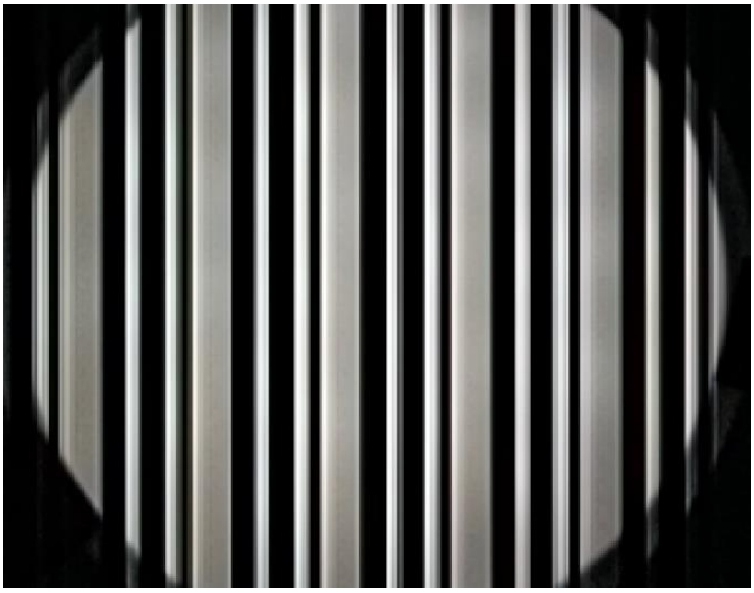
Hình 2. 10 Ảnh hưởng của tốc độ màn trập lên chuyển động

Nếu tốc độ màn trập đủ nhanh, ta có thể đóng băng được các chuyển động tạo ra hình ảnh chuyển động rõ nét. Ngoài ra, thời gian phơi sáng cũng ảnh hưởng đến độ nhiễu của bức ảnh. Đối với hệ thống truyền thông bằng ánh sáng, để có thể bắt

được các trạng thái của bóng đèn LED đang nhấp nháy, tốc độ màn trập cần phải được điều chỉnh về mức lớn nhằm loại bỏ nhiễu do ánh sáng bên ngoài lọt vào.

- **Tốc độ lăn (Shutter rate)**

Màn trập điện tử sử dụng cảm biến CMOS sẽ quét tất cả điểm ảnh theo chiều ngang hoặc chiều dọc. Trong hệ thống truyền thông bằng ánh sáng, chúng sẽ tạo ra những vạch màu xen kẽ khi ghi lại ảnh của đèn LED đang được bật tắt liên tục với tần số cao như hình minh họa bên dưới.



Hình 2. 11 Đèn LED được chụp bằng camera điện thoại dùng màn trập lăn

Trong camera điện thoại di động, tốc độ lăn là thông số rất quan trọng và nó được biểu diễn bằng giá trị Read – out time như phần trên đã trình bày. Giá trị Read – out time để tính số pixel tương ứng với một bit trong kỹ thuật truyền dữ liệu trong đề tài này. Thông số Read – out time không được cung cấp trong bất kỳ sản phẩm thương mại nào, do đó H. Nguyen và các cộng sự đã đề xuất một công thức tính sau [17]:

$$N_{pixel/bit} = \frac{T_{clock}}{T_{rolling}} \qquad PT\ 2.\ 2$$

trong đó: $N_{pixel/bit}$: là độ dày của 1 bit, T_{clock} là chu kỳ phát, $T_{rolling}$ là chu kỳ đọc của cảm biến.

Ta áp dụng phương pháp lấy mẫu Nyquist, nên có $T_{clock} \geq 2.T_{rolling}$. Như vậy, cần tiến hành thí nghiệm đo lường để xác định tốc độ lăn khi muốn biết độ dày của một bit.

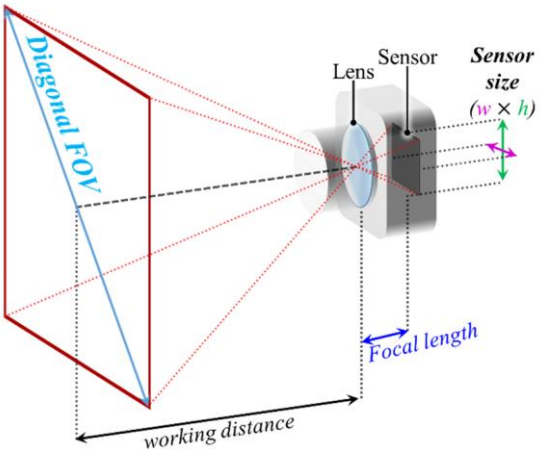
- **Độ dài tiêu cự**

Độ dài tiêu cự được định nghĩa là khoảng cách từ cảm biến hình ảnh và thấu kính. Độ dài tiêu cự ảnh hưởng đến kích thước của hình ảnh. Tiêu cự càng dài thì có thể cho ra ảnh có độ phóng đại càng lớn.

Mối quan hệ giữa độ dài tiêu cự và khoảng cách chụp được thể hiện trong phương trình sau:

$$\frac{d[m]}{H_{LED}[m]} = \frac{f[mm]}{h_{LED[mm]}} \qquad PT\ 2.\ 3$$

trong đó f là tiêu cự, d là khoảng cách chụp, $H_{LED}[m]$ là kích thước LED thực tế và h_{LED} là kích thước LED trong hình ảnh được chụp.



Hình 2. 12 Độ dài tiêu cự của camera

Từ PT 2. 3, ta có thể ước tính khoảng cách truyền tối đa trong VLC nếu biết được kích thước đèn LED, tiêu cự của máy ảnh và kích thước cảm biến. Về mặt lý thuyết, theo định lý lấy mẫu Nyquist khoảng cách lớn nhất được tính toán dựa trên kích thước của nguồn sáng trên hình ảnh sao cho là nhỏ nhất (2 pixel). Khi đó, ta có công thức sau:

$$d_{max}[m] = f[mm].H_{LED}[m].\frac{1}{2}.\frac{N\_pixel\_image\_row}{h\_sensor\_image[mm]} \qquad PT\ 2.\ 4$$

trong đó: $N\_pixel\_image\_row$ là số hàng pixel của một ảnh, h_sensor_image là chiều cao của cảm biến hình ảnh (milimet), d_{max} là khoảng cách lớn nhất có thể chỉ là lý thuyết vì trên thực tế khoảng cách còn phụ thuộc vào cường độ sáng. Nếu cường độ sáng của phần phát của VLC nhỏ hơn các nguồn sáng khác, hệ thống có thể sẽ bị ảnh hưởng.

- **Độ phân giải**

Độ phân giải là thông số thể hiện mức độ sắc nét của máy ảnh, các chi tiết rõ nét hay không. Đây là thông số để đánh giá chất lượng của camera có tốt hay không. Độ phân giải xác định chiều rộng và chiều dài của hình ảnh được chụp. Một triệu điểm ảnh tương đương với một megapixel, được viết tắt là MP (đôi khi là Mpix, Mpx hoặc Mpixel). Ví dụ độ phân giải camera sau và trước trên Google Pixel 4 lần lượt là: 16 MP và 8 MP. Trên điện thoại thông minh, độ phân giải của camera trước thường kém hơn camera sau nên việc lựa chọn sử dụng camera nào cũng cần được tính đến trong hệ thống VLC.

b. Phần mềm trên thiết bị di động

Trong hệ thống VLC này, em đã phát triển một phần mềm trên thiết bị di động cho phần thu và được xây dựng trên nền tảng Android Studio. Bảng 2. 4 giới thiệu về các thông số của điện thoại di động thực nghiệm.

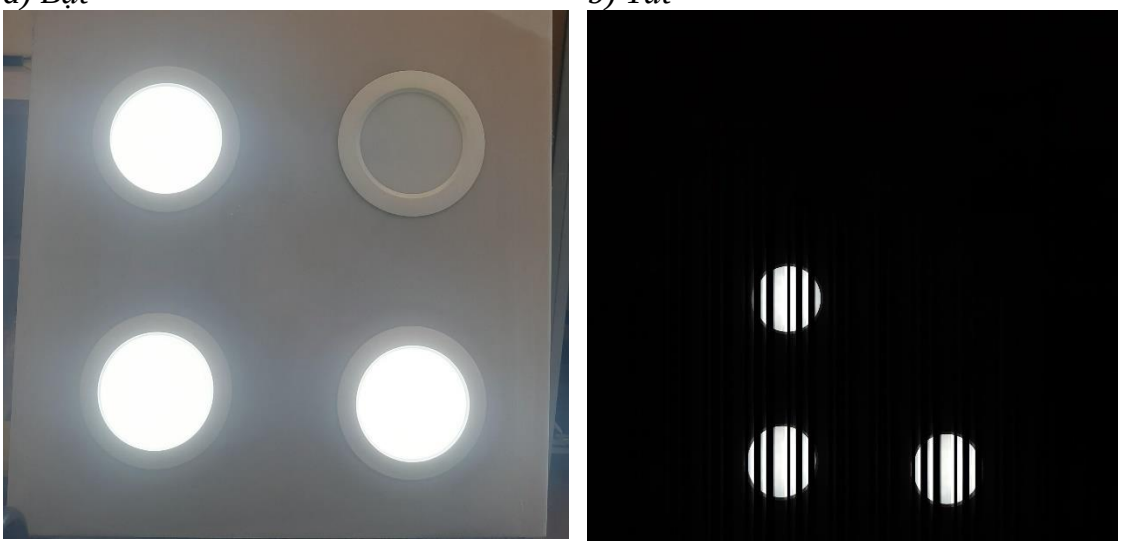
Tên thông số	Giá trị
Hãng điện thoại	Google Pixel 4
CPU	1 x 2.84GHz & 3 x 2.42GHz & 4 x 1.78GHz (Qualcomm Snapdragon 855)
GPU	257MHz
Cảm biến ảnh	Rolling Shutter CMOS
Tốc độ khung hình	30 fps
Camera	Camera trước: 8 megapixels Camera sau: 16 megapixels
Tiêu cự	28 mm – f/1.7 48 mm – f/2.4
Khẩu độ	f/1.7; f/2.4
Camera API	Camera2 API level 30
Độ phân giải hiển thị	1080 x 2220 px

Bảng 2. 4 Thông số điện thoại Google Pixel 4

Ngôn ngữ sử dụng trong việc xây dựng một phần mềm trên thiết bị di động: Python và Kotlin. Ngoài ra, em còn sử dụng các thư viện hỗ trợ sau: OpenCV, Numpy, Scipy và Math.

Một vài thiết lập trong ứng dụng này là tắt chế độ cân bằng ánh sáng tự động khi mở ứng dụng lên. Tắt chế độ cân bằng ánh sáng giúp cho vùng sáng của đèn LED được tách riêng với vùng sáng khác, giảm nhiễu trong quá trình giải mã.

a) Bật *b) Tắt*

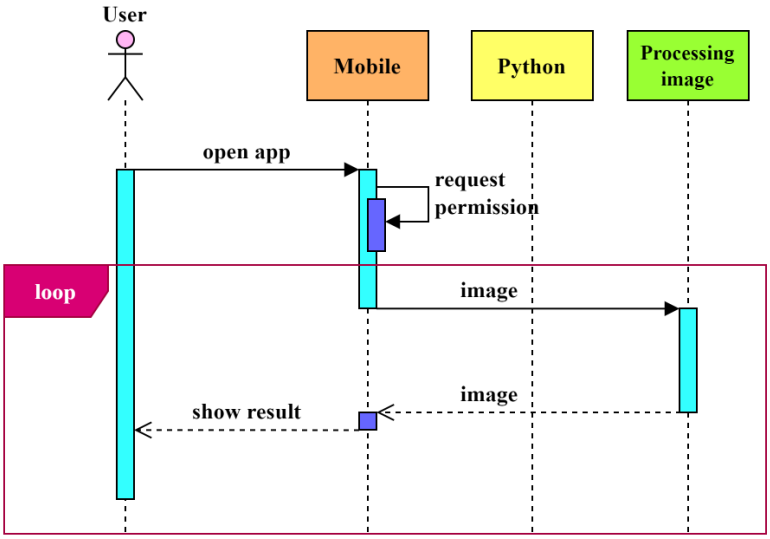


Hình 2. 13 Sự khác nhau khi bật và tắt chế độ cân bằng ánh sáng

2.2.3 Thuật toán xử lý dữ liệu đèn LED

a. Luồng xử lý của phần mềm trên thiết bị di động

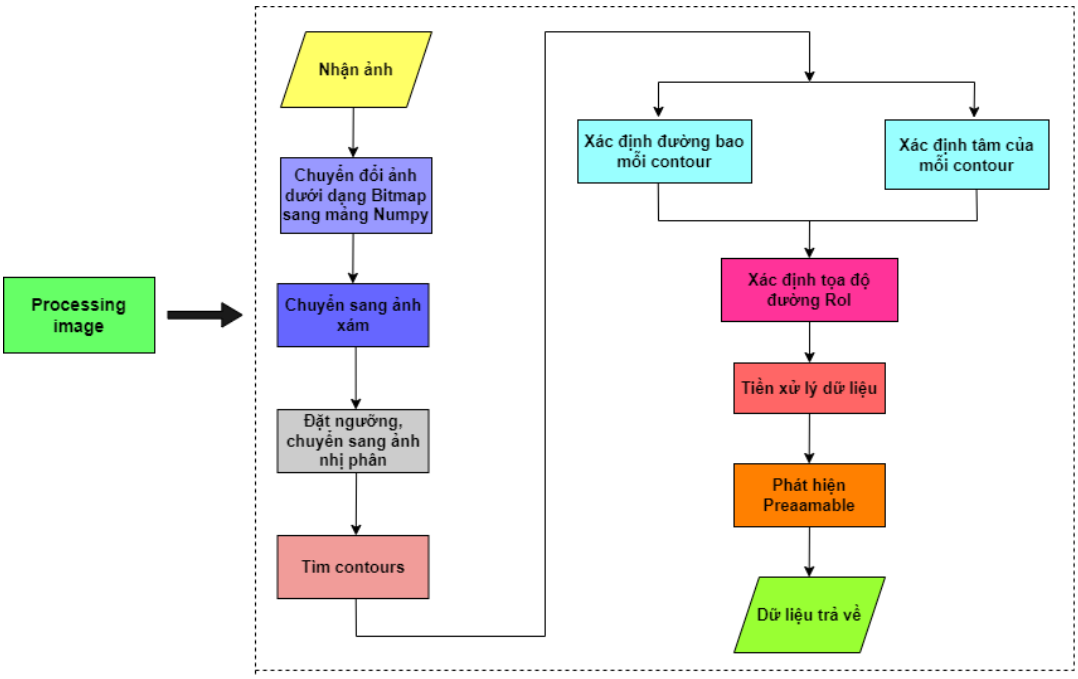
Hệ thống Rx là một phần mềm trên thiết bị di động được xây dựng trên nền tảng Android studio có luồng xử lý tuần tự như trong Hình 2. 14.



Hình 2. 14 Biểu đồ tuần tự của phần mềm

b. Thuật toán xử lý ảnh

Sơ đồ khối quá trình xử lý mỗi hình ảnh nhận được từ camera điện thoại thể hiện trong Hình 2. 15



Hình 2. 15 Sơ đồ khối quá trình xử lý mỗi hình ảnh

Để hiểu rõ hơn về quá trình trên, em sẽ phân tích từng bước:

- Chuyển đổi ảnh dưới định dạng bitmap (BMP) sang mảng Numpy: do bitmap là một định dạng hình ảnh kỹ thuật số được biểu diễn bằng các điểm

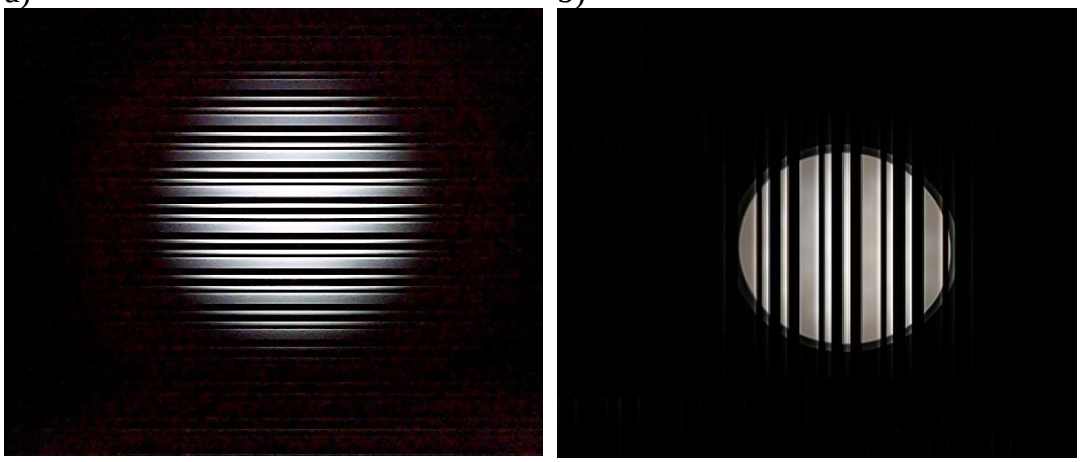
ảnh (pixel) và được lưu trữ trong bộ nhớ dưới dạng các byte. Mỗi điểm ảnh trong bitmap được biểu diễn bởi một byte (8 bit) hoặc nhiều byte tùy thuộc vào độ sâu màu của hình ảnh. Để thực hiện bước chuyển đổi này, thư viện Numpy đã hỗ trợ với câu lệnh:

```
img = np.frombuffer(frame, dtype=np.uint8).reshape(heigth,
width, 4)
```

- Xoay ảnh: trong quá trình thử nghiệm đề tài này với webcam thì cơ chế quét màn ảnh của webcam có khác so với điện thoại di động, cụ thể là webcam được thiết kế hiện thị hình ảnh theo chiều dọc của khung hình (Hình 2. 16) và thuật toán đã chạy thử nghiệm thành công trên webcam nên chỉ cần xoay ảnh ngược (hoặc cùng) kim đồng hồ 90 độ để thử nghiệm trên điện thoại di động trước khi xử lý phần tiếp theo. Việc xoay ảnh đã được thư viện OpenCV hỗ trợ:

```
img = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
```

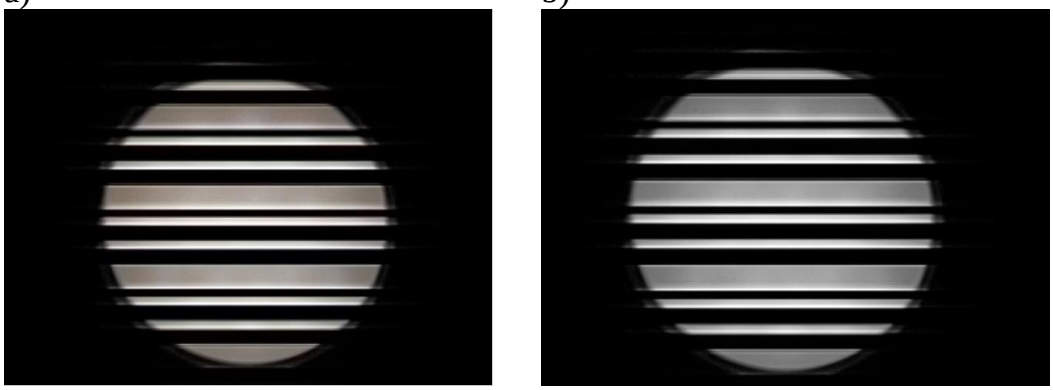
a) b)



Hình 2. 16: a) Hình ảnh nhận được bằng webcam; b) Hình ảnh nhận được bằng camera điện thoại di động

- Chuyển ảnh BGR (ảnh màu) sang ảnh xám Hình 2. 17.

a) b)



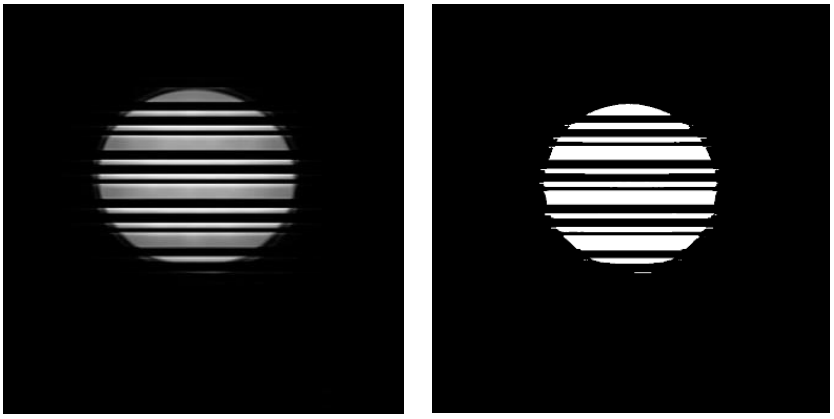
Hình 2. 17: a) Ảnh màu BGR; b) Ảnh xám

- Lựa chọn giá trị ngưỡng là để chuyển từ ảnh xám sang ảnh nhị phân. Việc hình ảnh trong quá trình xử lý đơn thuần chỉ là nguồn phát sáng, do đó em đã sử dụng phương pháp tính ngưỡng trung bình các giá trị có cường độ

sáng của mỗi pixel lớn hơn 10 (PT 2. 5) để tìm giá trị ngưỡng tự động toàn hình ảnh. Lý do chọn pixel lớn hơn 10 là để lọc trước các pixel có giá trị rất nhỏ (giá trị nhiễu) dẫn đến giá trị ngưỡng trung bình không được như mong muốn, ảnh hưởng đến quá trình xử lý sau đó. Những giá trị dưới ngưỡng trung bình sẽ trở thành giá trị 0 và trên ngưỡng trung bình sẽ trở thành giá trị 255 tương ứng với 0 và 1 trong ảnh nhị phân.

$$T_{threshold} = \frac{\sum_{j=0}^{N} I_j}{N} \qquad PT\ 2.\ 5$$

trong đó $T_{threshold}$ là giá trị ngưỡng cần tìm, I_j là giá trị cường độ sáng của mỗi điểm ảnh, N là số điểm ảnh có cường độ sáng lớn hơn 10.



Hình 2. 18: a) Ảnh xám; b) Ảnh nhị phân

- Từ hình ảnh nhị phân, áp dụng thuật toán tìm contours. Có thể hiểu contour (đường viền) là lấy tất cả các điểm liên tục (dọc theo đường biên) có cùng màu hoặc cường độ sáng. Một contour có thể chỉ là một điểm ảnh hoặc các điểm ảnh liên tiếp nhau. Mục đích tìm contours để ta phát hiện các đối tượng cần xử lý, chính là tìm các đối tượng màu trắng trên nền màu đen. Trong thư viện OpenCV, việc tìm và xử lý contours đều được hỗ trợ và cần thực hiện qua vài bước. Cụ thể trong đề tài này, để tìm contours dễ dàng thì ta sẽ sử dụng hình ảnh nhị phân (chỉ gồm hai màu đen và trắng) nên đã áp dụng threshold. Sau đó, ta chỉ cần sử dụng hàm sau để tìm các contours (Hình 2. 19):

```
contours, hierarchy = cv2.findContours(frame0, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

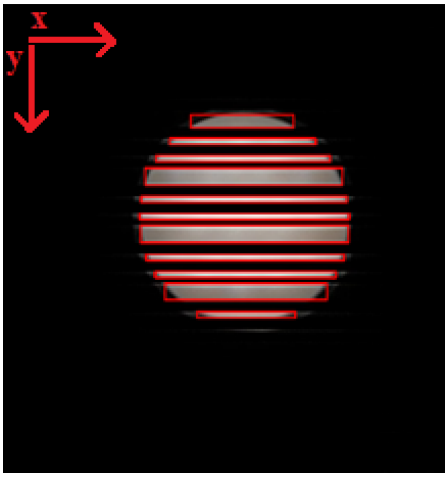
Có 3 tham số theo thứ tự trong hàm *cv2.findContours()*: hình ảnh gốc, phương pháp trích xuất contours, phương pháp xấp xỉ contour. Kết quả trả lại là hierarchy và danh sách contour. Trong đó, danh sách contours là toàn bộ các contours xác định trong hình ảnh. Mỗi một contour là một Numpy array của các tọa độ (x, y) của các điểm biên trong đối tượng. Còn hierarchy là một mảng đa chiều chứa thông tin liên quan đến các mối quan hệ giữa các đường contour.

Hình 2. 19 Ảnh vẽ contours tìm được

Hình ảnh trên có tổng là *27 contours*. Mảng chứa danh sách contour sẽ lưu các contour theo thứ tự từ dưới lên và từ trái sang phải của ảnh nhị phân. Trường hợp các contour có cấp bậc với nhau thì sẽ lưu lớp cha trước rồi đến các lớp con.

- Sau khi có danh sách contours, ta sẽ vẽ được hình chữ nhật bao quanh một đường contour (Hình 2. 20) bằng hàm *cv2.boundingRect(contour)* trong thư viện OpenCV. Hàm này có tham số đầu vào là một contour đã được tìm thấy trước đó. Kết quả trả về là một tuple gồm 4 giá trị: x, y, w, h.
    - x: tọa độ x của góc trên bên trái của hình chữ nhật bao quanh contour.
    - y: tọa độ y của góc trên bên trái của hình chữ nhật bao quanh contour.
    - w: chiều dài của hình chữ nhật bao quanh contour.
    - h: chiều rộng của hình chữ nhật bao quanh contour.

  Việc vẽ hình chữ nhật bao quanh một contour để tạo ra một khung hình chính xác và hiệu quả để bao quanh đối tượng trong hình ảnh. Từ đó, cung cấp thông tin về tọa độ y của cạnh trên và cạnh dưới hình chữ nhật để xử lý phần tiếp theo.



Hình 2. 20 Hình chữ nhật bao quan mỗi contour

- Ngoài việc xác định đường bao mỗi contour, ta còn tính được tâm của mỗi contour sau khi có danh sách contour. Thuật toán tìm tâm của mỗi contour trong thư viện OpenCV sẽ liên quan đến một khái niệm gọi là moment.

Moment là một đại lượng toán học được sử dụng để mô tả các thuộc tính hình học của một đối tượng trong ảnh bao gồm tọa độ trọng tâm, diện tích và hướng. Giá trị mỗi moment (M_{ij}) đại diện cho phân phối không gian của các pixel trong một đối tượng. Cụ thể như sau:

$$M_{ij} = \sum_i \sum_j x^i y^j I(x,y) \qquad \textit{PT 2. 6}$$

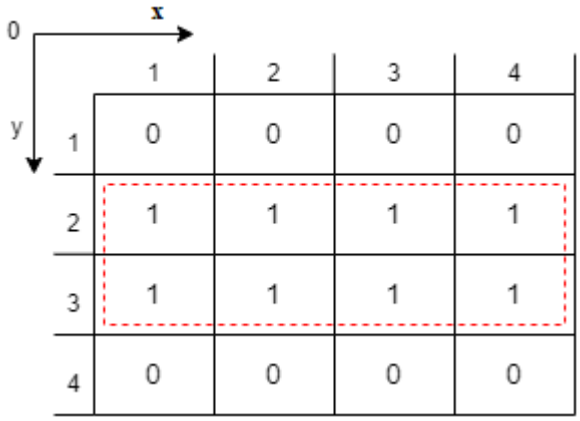
Từ PT 2. 6, M_{00} được xác định:

$$M_{00} = \sum_i \sum_j I(x,y) \qquad \textit{PT 2. 7}$$

Trong trường hợp ảnh nhị phân, cường độ sáng chỉ có hai giá trị duy nhất là 0 và 1. Điều này tương ứng với việc đếm số lượng pixel có giá trị khác không (1) sẽ tương đương với diện tích của vùng đó trong ảnh. Vì chỉ có hai giá trị cường độ sáng, moment M_{00}của ảnh nhị phân sẽ cho kết quả là diện tích của vùng đó. Moment M_{10} và M_{01} trong contour được sử dụng để tính toán trọng tâm của đường viền. Do đó, tâm được xác định bởi công thức sau:

$$\{\bar{x}, \bar{y}\} = \{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\} \qquad \textit{PT 2. 8}$$

Xét một ví dụ đơn giản với một ảnh nhị phân 4x4, trong đó có một contour là vùng khoanh đỏ:



Theo PT 2. 7:

$$M_{00} = số\ lượng\ các\ pixel\ khác\ 0 = 8$$

Theo PT 2. 8, tọa độ tâm $\bar{x}$ được tính:

$$\bar{x} = \frac{M_{10}}{M_{00}} = \frac{\sum_x \sum_y x\, I(x,y)}{\sum_x \sum_y I(x,y)}$$

$$= \frac{2\times1 + 2\times1 + 2\times1 + 2\times1 + 3\times1 + 3\times1 + 3\times1 + 3\times1}{8} = \frac{20}{8}$$

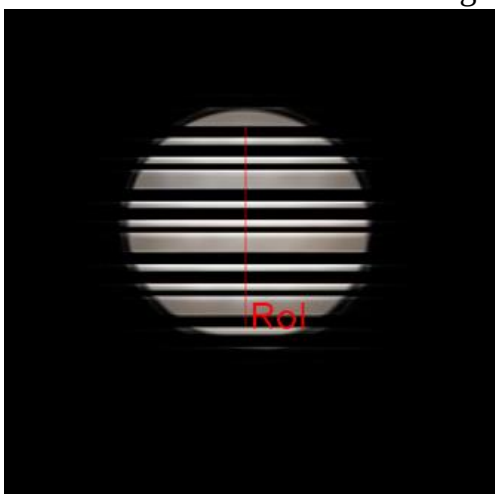
tọa độ tâm $\bar{y}$ được tính:

$$\bar{y} = \frac{M_{01}}{M_{00}} = \frac{\sum_x \sum_y y\, I(x,y)}{\sum_x \sum_y I(x,y)}$$

$$= \frac{1\times 1 + 1\times 2 + 1\times 3 + 1\times 4 + 1\times 1 + 1\times 2 + 1\times 3 + 1\times 4}{8} = \frac{20}{8}$$

Cuối cùng ta có được tâm của một contour với tọa độ $\{\bar{x}, \bar{y}\} = \{\frac{20}{8}, \frac{20}{8}\}$. Với bài toán xử lý đèn LED, thì ta sẽ xác định các tâm contour của từng sọc sáng. Cuối cùng, ta sẽ có một mảng chứa các tọa độ tâm bao gồm mảng tọa độ x và mảng tọa độ y.

- Xác định RoI: mỗi hình ảnh ta cần phải tìm một vùng quan tâm để tập trung phân tích, xử lý và thuật ngữ dùng để miêu tả điều này gọi là **RoI** (Region of Interest). RoI có thể là một khu vực hình chữ nhật, hình vuông, một đường thẳng hoặc hình dạng khác. Việc xác định và xử lý chỉ trong vùng RoI để giảm thiểu thời gian tính toán và tăng độ chính xác của quá trình xử lý ảnh. Dưới đây là hình ảnh đèn LED có chứa đường RoI.



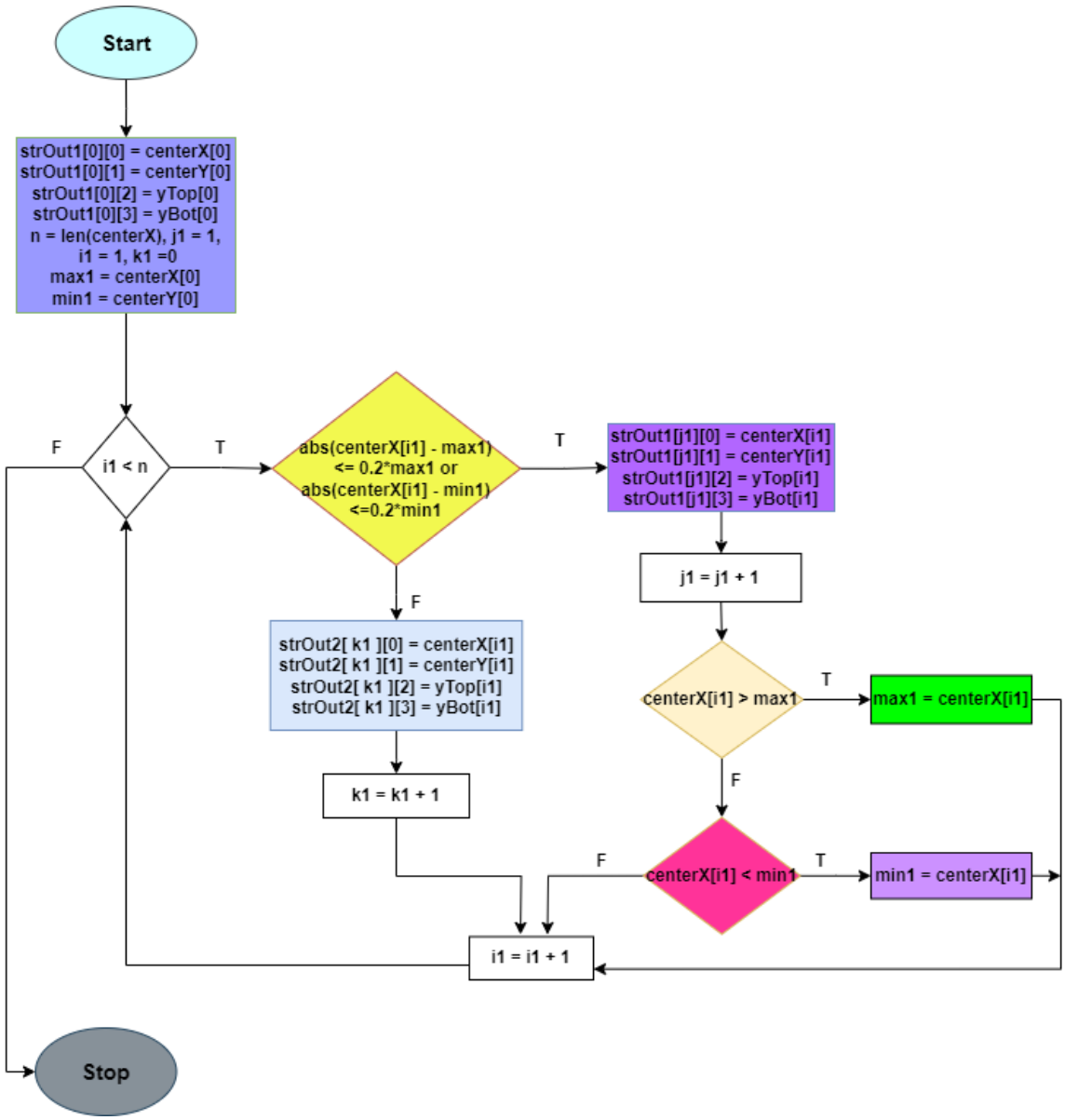
Hình 2. 21 Ảnh chứa RoI

RoI trên hình là một đường thẳng đi qua những vùng có cường độ sáng tốt nhất. Để có được đường RoI này quy trình xử lý sẽ như sau:

Hình 2. 22 Sơ đồ khối quá trình xác định tọa độ đường RoI

Hai giá trị quan trọng nhất để có được đường RoI chính là tọa độ điểm đầu và điểm cuối của đường RoI. Để có được 2 tọa độ này, bước đầu tiên ta cần thuật toán so sánh tâm của các contour để lọc ra được các mảng contours phù hợp cho đường RoI đi qua. Điều kiện để lọc các mảng contours là ta đã có 4 mảng giá trị dưới đây:

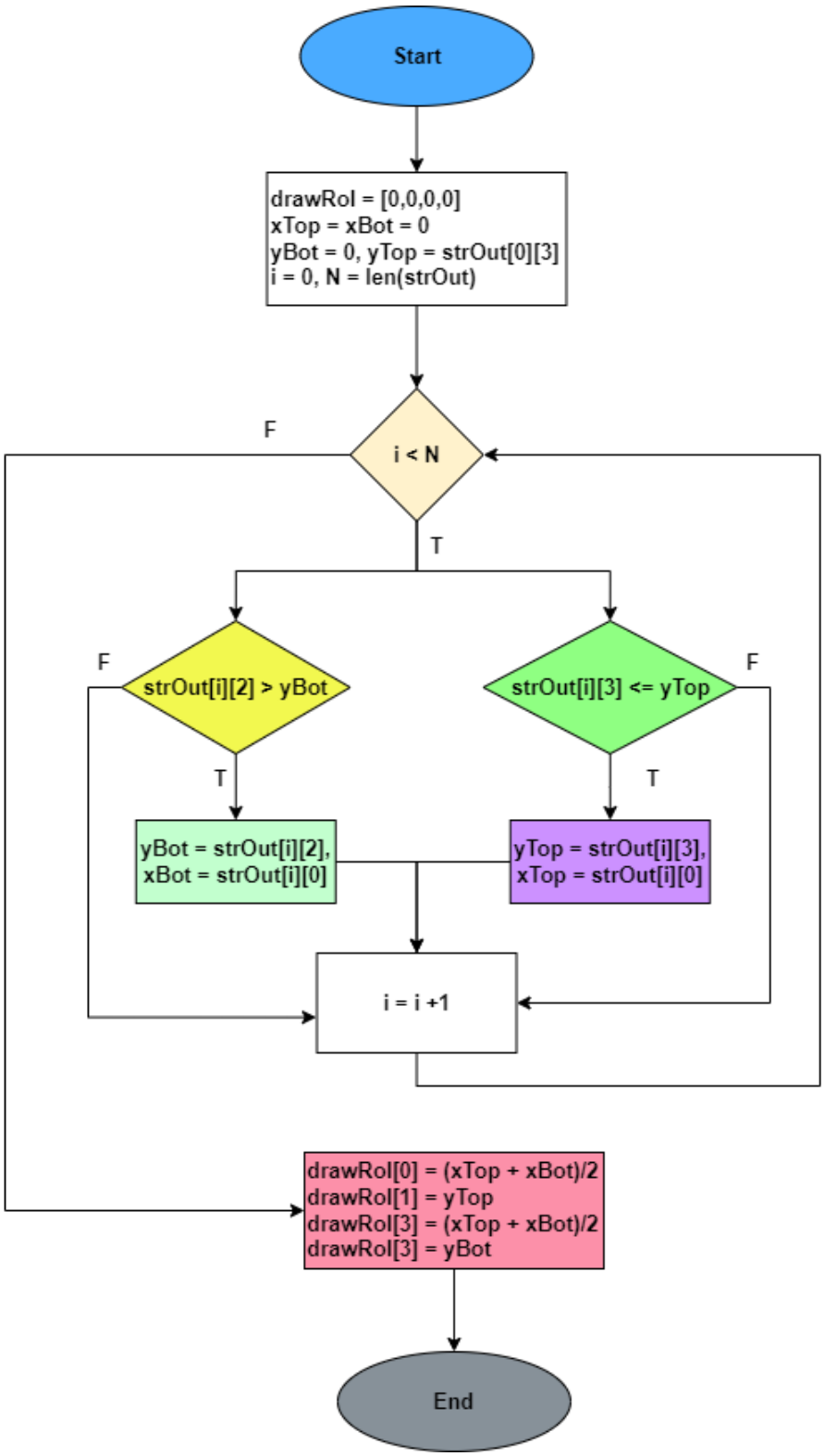
- mảng một chiều tọa độ x của tâm contour: centerX [ ]
- mảng một chiều tọa độ y của tâm contour centerY [ ]
- mảng một chiều tọa độ y của cạnh trên của đường bao: yTop [ ]
- mảng một chiều tọa độ y của cạnh dưới của đường bao: yBot [ ]



Hình 2. 23 Lưu đồ thuật toán lọc ra 4 mảng contours sử dụng so sánh tâm contour.

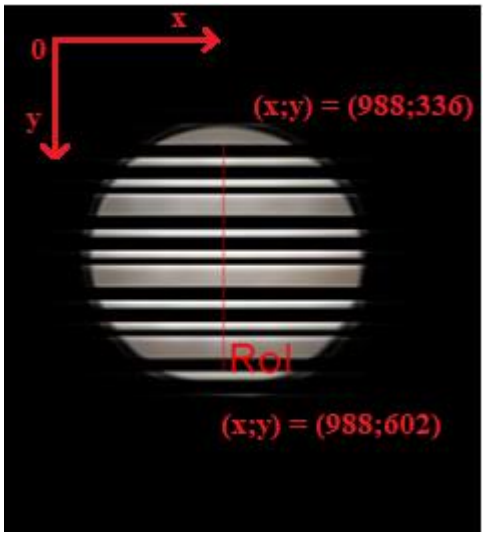
Lưu đồ thuật toán trên sẽ được thực hiện 4 lần là để lọc ra được 4 mảng contour sao cho mỗi mảng chứa các tâm contour thỏa mãn điều kiện như trên. Việc lọc ra 4 mảng là do thí nghiệm tối đa với 4 đèn LED, vì vậy sẽ cần tối thiểu 4 đường RoI. Kết quả 4 mảng trên sẽ sử dụng lần lượt là: strOut1, strOut3, strOut5, strOut7. Mỗi mảng là mảng 2 chiều với kích thước n x 4 (n: hàng, 4: cột), giá trị các cột theo thứ tự lần lượt là: centerX, centerY, yTop, yBot.

Sau khi có 4 mảng, ta sẽ sắp xếp kích thước của 4 mảng để quá trình tìm vị trí điểm đầu, điểm cuối đường RoI sẽ xử lý từ mảng có kích thước lớn nhất tới nhỏ nhất. Điều này giúp đường RoI tương ứng với đèn LED của nó chính xác hơn. Lưu đồ thuật toán dưới đây sẽ tìm vị trí điểm đầu, điểm cuối của đường RoI:



Hình 2. 24 Lưu đồ thuật toán tìm tọa độ điểm đầu và điểm cuối của đường RoI

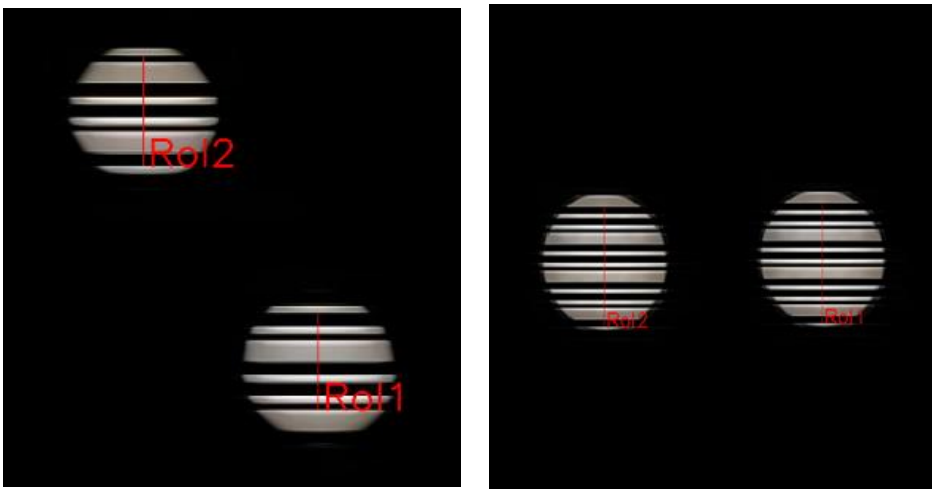
Kết quả của quá trình trên sẽ là mảng chứa tọa độ vị trí điểm đầu, điểm cuối của đường RoI như trong hình bên dưới.



Hình 2. 25 Tọa độ điểm đầu và điểm cuối của 1 đường RoI

Việc lấy tọa độ xTop, xBot của đường RoI bằng trung bình cộng của 2 giá trị đó và tọa độ yTop, yBot trực tiếp như vậy với mục đích làm cho đường RoI sẽ là một đường thẳng khi nối hai điểm với nhau, giúp cho đường RoI đi qua vùng sáng nhất của bóng đèn với tỉ lệ cao hơn. Nhưng trong quá trình thực nghiệm với 2 đèn trở lên thì điều này sẽ bị ảnh hưởng nếu ở các góc và hướng khác nhau giữa camera với cụm đèn LED.

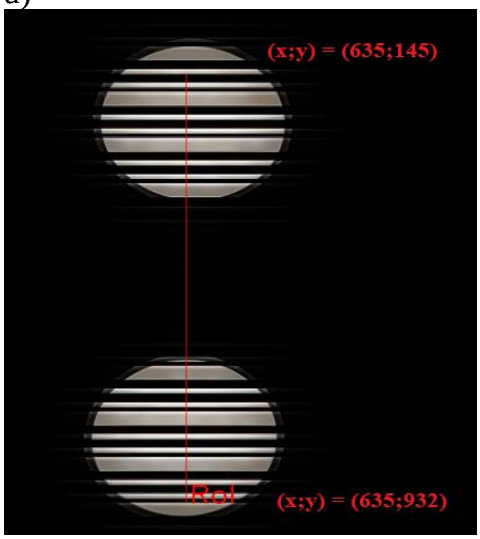
- Trường hợp 1 – cụm 2 đèn LED: góc giữa điện thoại và cụm LED ở vị trí mà có 2 RoI độc lập:



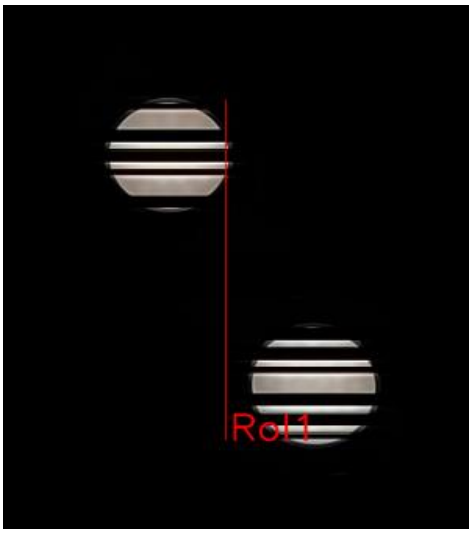
Hình 2. 26 Vị trí đèn và camera có 2 RoI độc lập

- Trường hợp 2 – cụm 2 LED: góc giữa điện thoại và cụm LED chỉ ra được 1 RoI:

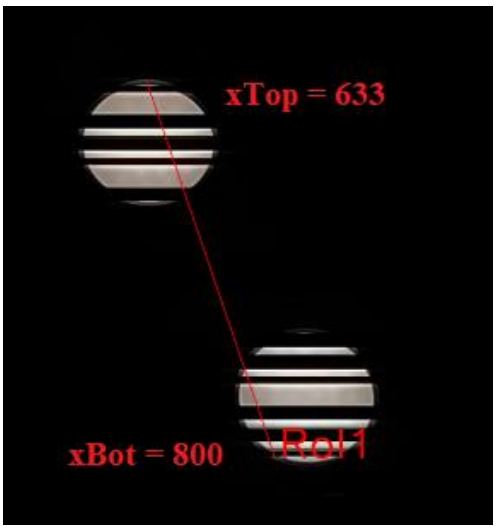
a)



b)



c)



Hình 2. 27: a) và b) Lấy giá trị trung bình tọa độ x; c) Không lấy giá trị trung bình tọa độ x

Cả 3 hình ảnh trên chỉ có một đường RoI mặc dù có hai đèn LED. Để có được các đường RoI độc lập tương ứng với các đèn LED thì ta sẽ tính độ dài và góc lệch của đường RoI gốc so với phương thẳng đứng:

$$D = |yTop - yBot| \qquad PT\ 2.\ 9$$

$$\alpha = \tan^{-1}\frac{|xBot - xTop|}{D} \qquad PT\ 2.\ 10$$

❖ Nếu $D \geq 35 \times N_{pixel/bit}$, $\alpha \leq 2°$ thì:

$$xTop1 = xBot1 = xTop2 = xBot2 = \frac{xTop + xBot}{2}$$

$$yTop1 = yTop;\ yBot1 = yTop + 20 \times N_{pixel/bit}$$

$$yTop2 = yBot - 20 \times N_{pixel/bit};\ yBot2 = yBot$$

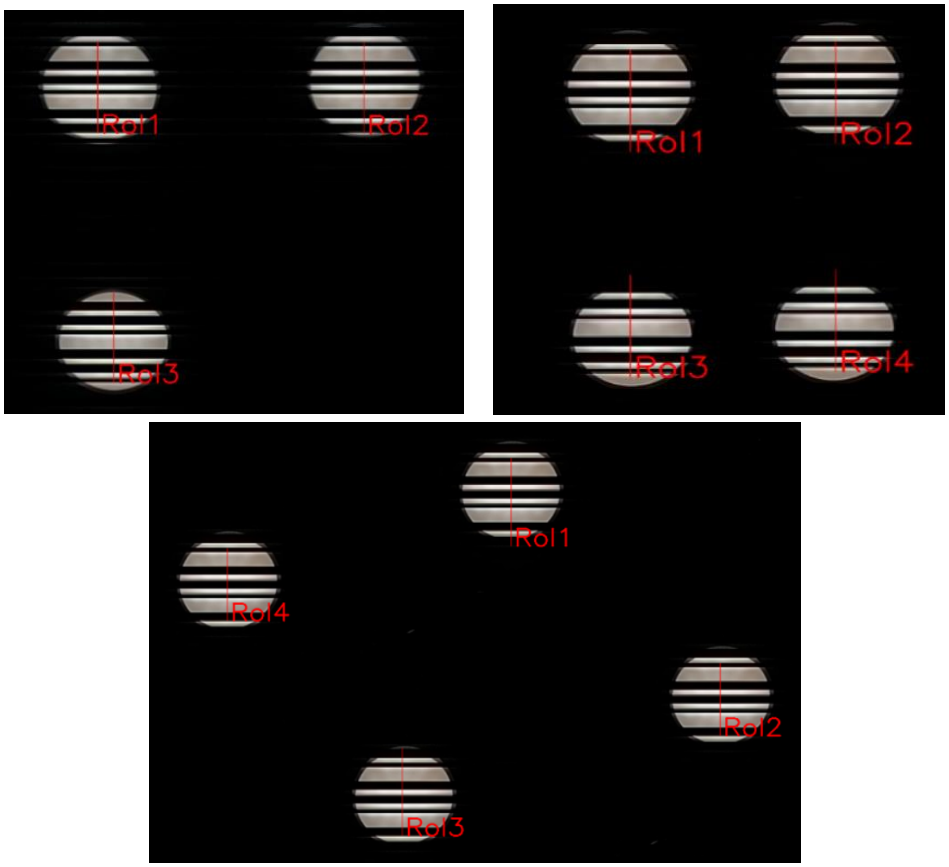
❖ Nếu $D \geq 35 \times N_{pixel/bit}$, $\alpha > 2°$ thì:

$$xTop1 = xBot1 = xTop;\ xTop2 = xBot2 = xBot$$

$$yTop1 = yTop;\ yBot1 = yTop + 20 \times N_{pixel/bit}$$
$$yTop2 = yBot - 20 \times N_{pixel/bit};\ yBot2 = yBot$$

trong đó: D là khoảng cách giữa điểm đầu và điểm cuối của đường RoI ban đầu, α là góc lệch đường RoI so với phương thẳng đứng.

Giá trị $N_{pixel/bit}$ sẽ có phương pháp tính riêng ở phần **2.2.4 (Tính độ dày của 1 bit)**. Với cách tính trên, ta có thể áp dụng cho cụm 3 hoặc 4 đèn LED đều ra được các đường RoI độc lập dù đứng ở các góc, hướng khác nhau giữa camera và LED.



Hình 2. 28 Sau khi tách RoI

- Tiền xử lý dữ liệu: ở bước này hàm sẽ xử lý mảng các giá trị pixel trên đường RoI đi qua ảnh xám.

  Sau khi có đường RoI, ta sẽ đếm được số sọc sáng (tối) mà đường RoI đi qua bằng quan sát và đo được độ dài của của RoI. Từ đó, ta xác định số pixel trên một sọc sáng (tối) nhỏ nhất tương đương với 1 bit (0 hoặc 1) hay chính là $\boldsymbol{N_{pixel/bit}}$. Sau đó, ta thực hiện lấy một mảng chứa các giá trị điểm ảnh trên ảnh xám với khoảng cách $\boldsymbol{N_{pixel/bit}}$ mà đường RoI đi qua. Mảng giá trị này được đưa qua hàm **Curve fit** để xử lý. Sử dụng hàm Curve fit để tối ưu việc chuyển từ ảnh xám sang ảnh nhị phân so với cách lựa chọn ngưỡng như ban đầu trong khi việc phân tích đưa ra dữ liệu truyền cần độ chính xác cao.

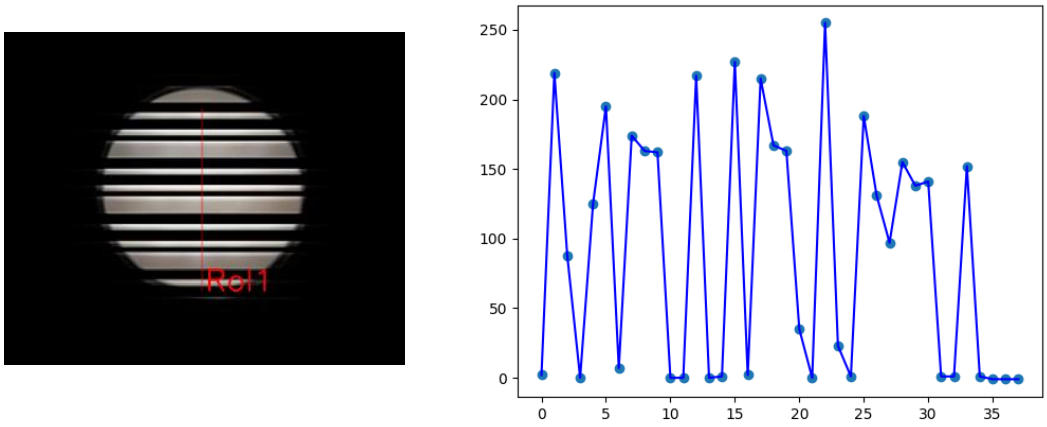
  Hàm Curve fit sẽ tìm các tham số tối ưu cho một mô hình toán học dựa trên dữ liệu có sẵn. Mục tiêu của Curve fit là xấp xỉ mô hình dự đoán sao cho

nó gần với dữ liệu thực tế nhất có thể. Mô hình dự đoán được biểu diễn như sau:

$$y_{data} = f(x_{data}, params) + eps \qquad PT\ 2.\ 11$$

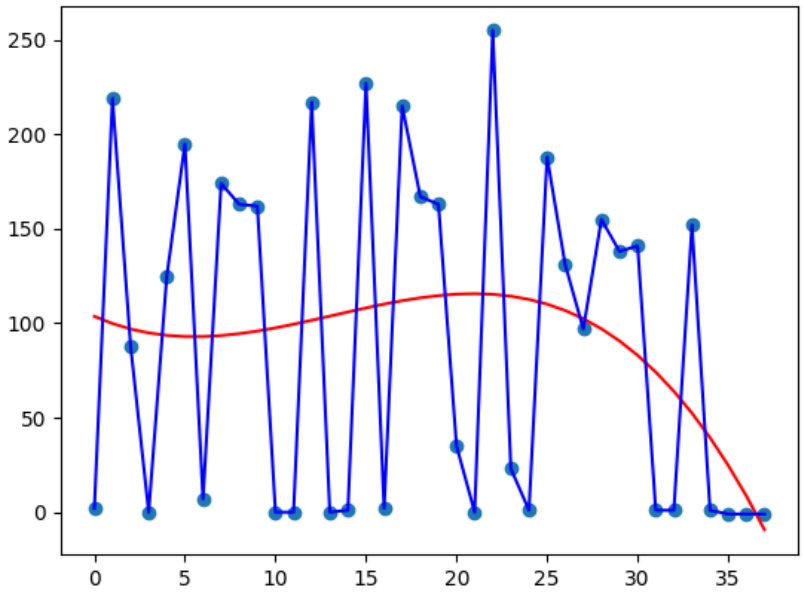
trong đó:

- $\boldsymbol{f}$: là một hàm mô tả mô hình dự đoán. Nó ánh xạ từ dữ liệu đầu vào **x_data** và các tham số params sang dữ liệu đầu ra **y_data**. Hàm curve fit sẽ tìm cách tối ưu hóa các tham số params sao cho giá trị dự đoán của f gần với dữ liệu thực tế nhất có thể.
- **params**: là cú pháp sử dụng để truyền một danh sách các tham số vào hàm f
- **eps**: là thành phần nhiễu, thường được giả định là một phân phối ngẫu nhiên có thể gây ra sai số trong dữ liệu thực tế.



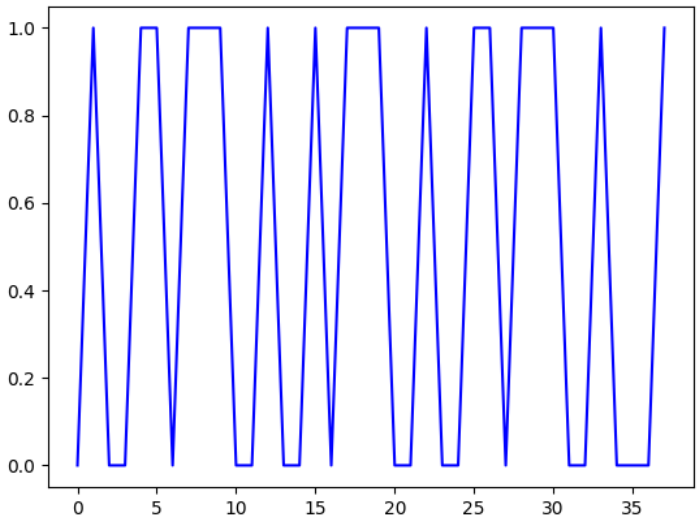
Hình 2. 29 Giá trị các điểm ảnh đường RoI đi qua trên ảnh xám cách nhau $N_{pixel/bit}$.

Hình 2. 29 trên là biểu đồ thể hiện các giá trị điểm ảnh trên ảnh xám với khoảng cách $\boldsymbol{N_{pixel/bit}}$ mà đường RoI đi qua. Trục ngang là chỉ số tương ứng với giá trị mỗi pixel trên trục dọc. Áp dụng hàm Curve fit ta sẽ được một đường cong như Hình 2.30



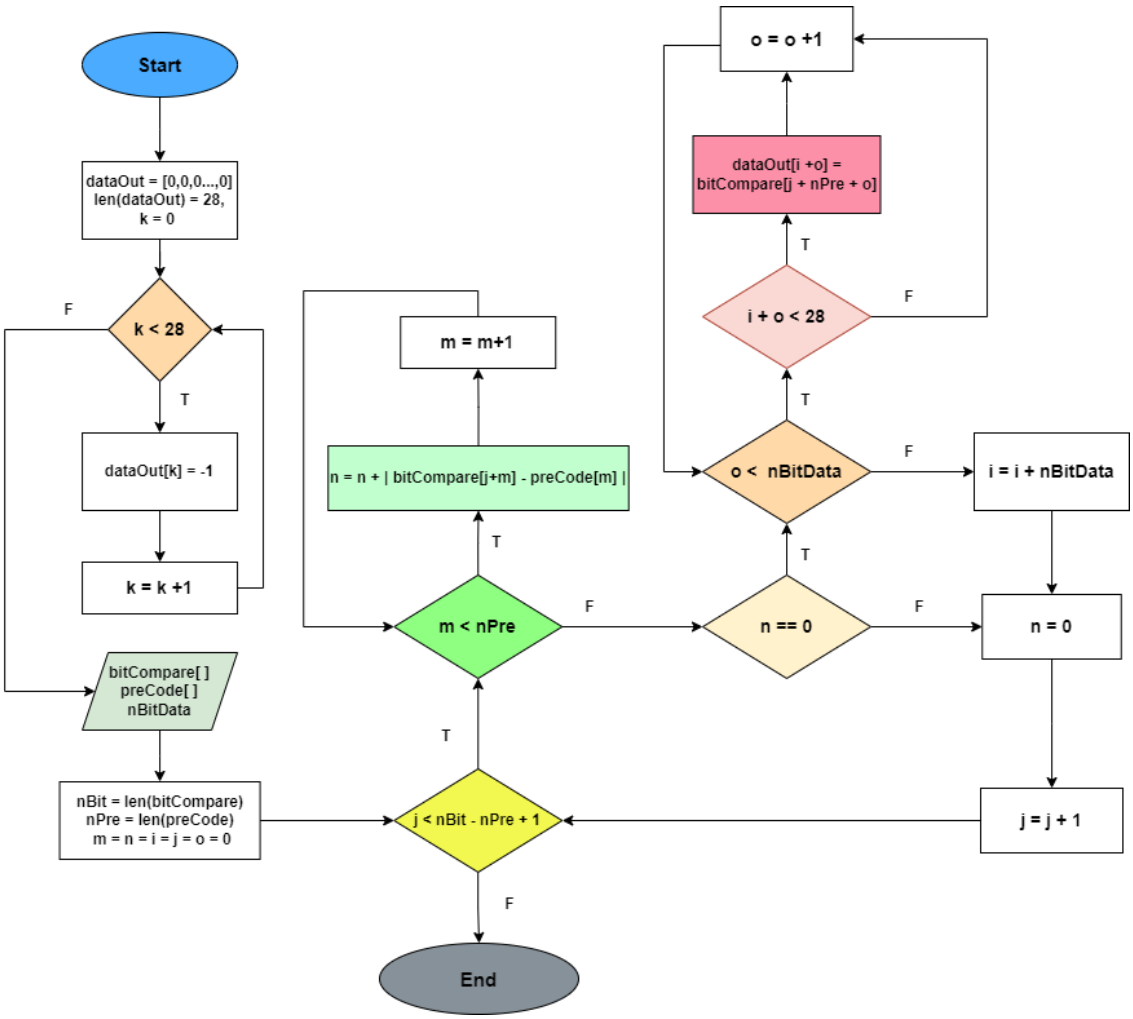
Hình 2. 30 Đường curve fit có màu đỏ

Những giá trị nằm trên đường cong sẽ nhận giá trị 1 và nằm dưới sẽ nhận giá trị 0. Ta nhận được một mảng giá trị 0 và 1 để có thể giải mã ra dữ liệu truyền.



Hình 2. 31 Đồ thị giá trị 0 và 1 từ đường curve fit.

- Sau khi có được mảng dãy giá trị 0 và 1, ta so sánh mảng đó với 6 bits premable truyền đến nếu đúng thì sẽ lấy 4 hoặc 8 bits payload tiếp theo. Lưu đồ thuật toán thực hiện quá trình này ở trong Hình 2. 32

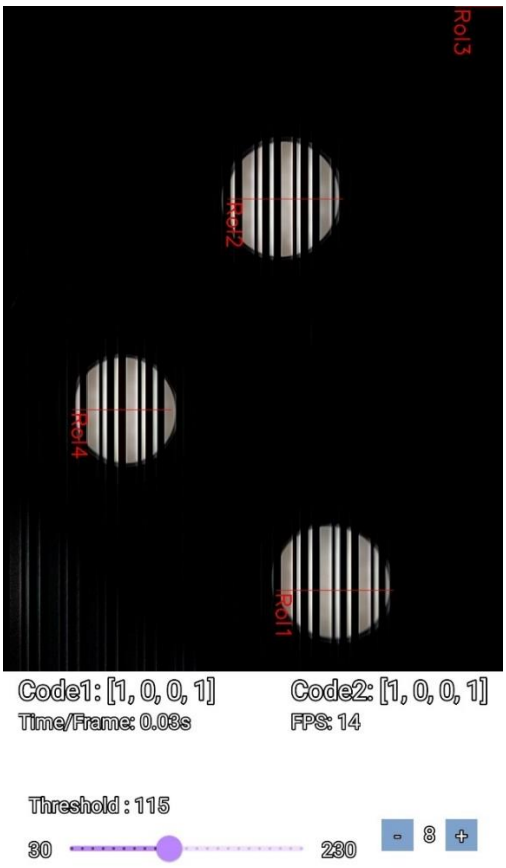


Hình 2. 32 Lưu đồ thuật toán giải mã ra 4 hoặc 8 bits payload

Giải thích cách đặt tên biến:

- ❖ dataOut: mảng chứa tất cả giá trị dữ liệu đã được giải mã
- ❖ bitCompare: mảng bit 0 và 1 nhận được sau khi sử dụng Curve fit.
- ❖ preCode: 6 bit premable
- ❖ nBitData: số bit dữ liệu cần giải mã (4 hoặc 8)

- Trong mảng dataOut sẽ có những nBitData đã được giải mã do một ảnh sẽ có thể nhận được nhiều frame dữ liệu truyền đến. Vì vậy, ta cần lọc xem nBitData nào sẽ xuất hiện nhiều nhất trong mảng thì đó chính là nBitData cuối cùng cần tìm và trả kết quả về, kết thúc quá trình xử lý một ảnh.



Hình 2. 33 Kết quả trả về toàn bộ quá trình xử lý 1 hình ảnh

Toàn bộ quá trình trên sẽ được lặp lại liên tục với từng ảnh nhận được. Mỗi lần kết quả trả về có thể lưu lại để tính sai số dữ liệu đã được giải mã trong 100 hình ảnh đã xử lý được.

2.2.4 Tính độ dày của 1 bit

Độ dày của 1 bit là chiều rộng của một sọc sáng hoặc tối (dãy điểm ảnh) tương ứng với một bit có giá trị 1 hoặc 0.

a. Phương pháp thủ công

Từ PT 2. 2 ta có công thức tính tốc độ lăn $f_{rolling}$ là:

$$f_{rolling} = N_{pixel/bit} \cdot f_{clock} \qquad \textit{PT 2. 12}$$

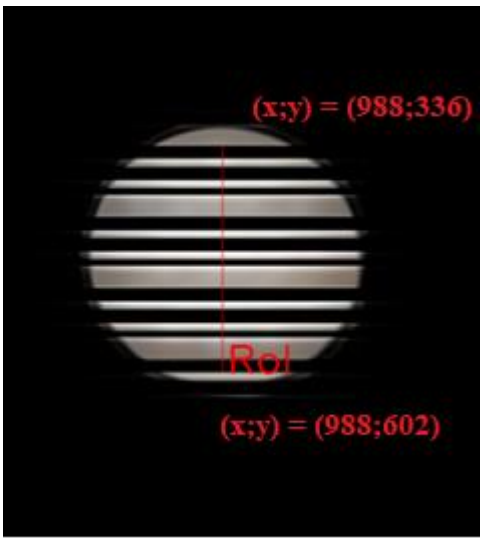
trong đó $f_{rolling}$ là tốc độ lăn, $N_{pixel/bit}$ là độ dày của 1 bit, f_{clock} là tần số truyền.

Thông số này không được các nhà sản xuất điện thoại thông minh cung cấp về camera, nên ta cần tính toán thông số này đầu tiên. Cách thức thực hiện là phát xung có tần số cố định (ở đây lựa chọn 1000Hz), sau đó đếm số pixel tương ứng tại Rx.

Để tính độ dày của mỗi sọc đen, trắng tương đương với một bit 0 hoặc 1, ta dùng công thức sau:

$$N_{pixel/bit} = \frac{D}{n_{sọc}} \qquad \textit{PT 2. 13}$$

trong đó: D là độ dài của RoI, $n_{sọc}$ là số sọc đếm được trên đường RoI.



Hình 2. 34 Tọa độ đường RoI

Từ Hình 2. 34, sau khi chạy chương trình để tìm tọa độ điểm đầu và điểm cuối của đường RoI, ta nhận được hai tọa độ là (988,336) và (988,602). Sau đó, ta tính được D = 602 – 336 = 266 (pixel), $n_{sọc} = 32$

Do đó:

$$N_{pixel/bit} = \frac{266}{32} = 8.3$$

Cuối cùng ta tính được tốc độ lăn của camera điện thoại sử dụng là:

$$f_{rolling} = 8{,}3.1000 = 8300\ (Hz)$$

$N_{pixel/bit}$ là một thông số quan trọng quyết định đến sai số trong quá trình truyền nhận. Với mỗi tần số khác nhau, $N_{pixel/bit}$ cần phải được tính toán và truyền vào chương trình.

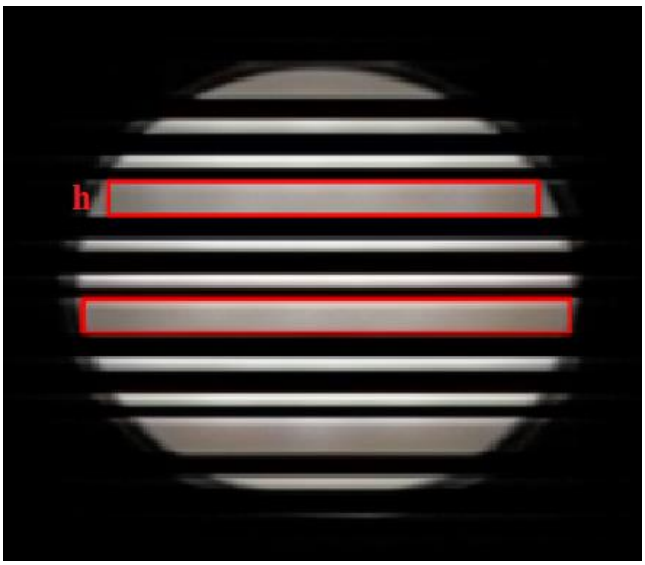
b. Phương pháp tự động

Khi áp dụng phương pháp tính thủ công cho giá trị $N_{pixel/bit}$ sẽ làm hạn chế đi việc ứng dụng trên nhiều dòng thiết bị di động do mỗi camera trên các thiết bị di động sẽ có tốc độ lăn khác nhau. Từ đó, ta cần một phương pháp mới để tự động tính toán giá trị $N_{pixel/bit}$ theo từng loại máy.

Như em đã trình bày ở phần **Cấu trúc gói dữ liệu**, với khung bản tin truyền dữ liệu hiện tại thì sọc sáng có độ dày lớn nhất sẽ mang 3 bits có giá trị 1 (Hình 2. 35). Sau đó, ta tính được chiều rộng h của từng sọc sáng qua việc sử dụng vẽ hình chữ nhật bao quanh contour (phần b). Từ đó, giá trị $N_{pixel/bit}$ được tính như sau:

$$N_{pixel/bit} = \frac{h_{max}}{3} \qquad \textit{PT 2. 14}$$

trong đó: $N_{pixel/bit}$ là độ dày của 1 bit (lấy giá trị nguyên); h_{max} là chiều rộng có giá trị lớn nhất trong tất cả hình bao.



Hình 2. 35 Sọc sáng có độ dày lớn nhất.

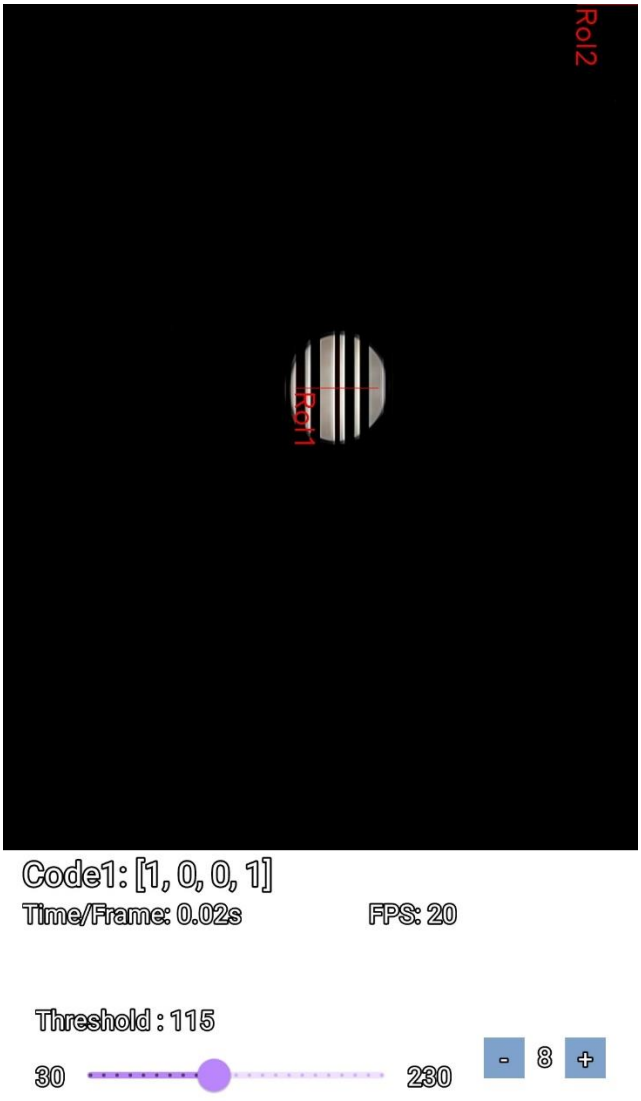
CHƯƠNG 3. Thực nghiệm và kết quả

3.1 Thực nghiệm

Thực nghiệm số 1, 2, 3 là với đèn LED tròn; thực nghiệm số 4 là với đèn LED panel.

3.1.1 Thực nghiệm số 1

a. **Trường hợp 1.1**: thực hiện truyền dữ liệu trên 1 đèn với chuỗi 10 bits "011100 1001" (4 bits dữ liệu: 1001) ở tần số 1000Hz. Hình ảnh của đèn LED thực nghiệm được hiển thị dưới đây.



Hình 3. 1 Thực nghiệm trường hợp 1.1

Từ đó ta thu được số liệu trong Bảng 3. 1:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác (%)
30	98
40	93
50	96
60	97
70	87
80	59

Bảng 3. 1 Giá trị thực nghiệm trường hợp 1.1

b. **Trường hợp 1.2:** đèn LED bên trái tiếp tục được sử dụng làm đèn truyền dữ liệu vẫn với chuỗi 10 bits "011100 1001" ở tần số 1000 Hz, tuy nhiên ở đèn LED bên phải bật sáng liên tục làm nhiễu, hình ảnh đèn như Hình 3. 2

Hình 3. 2 Thực nghiệm trường hợp 1.2

Từ đó ta thu được số liệu trong Bảng 3. 2:

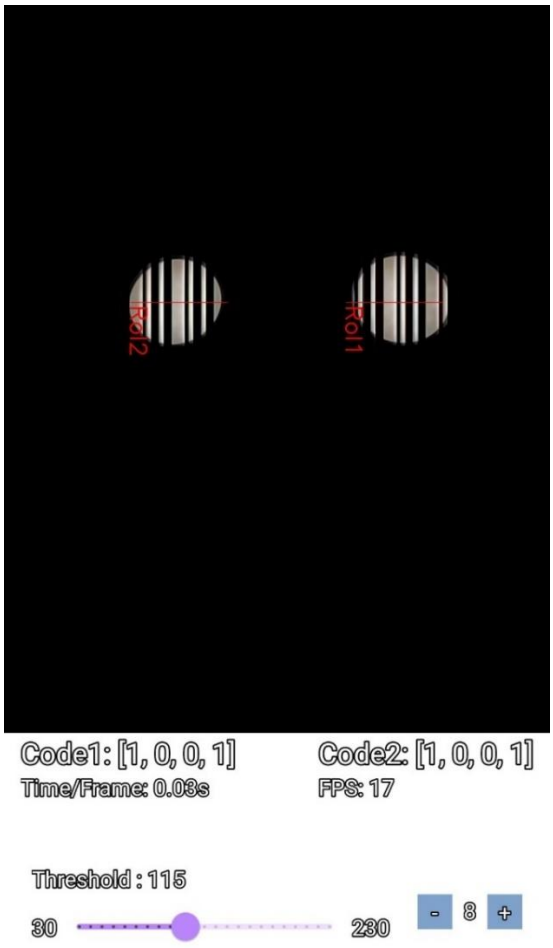
Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 1 LED (%)
30	93
40	94
50	94
60	90
70	84
80	60

Bảng 3. 2 Giá trị thực nghiệm trường hợp 1.2

3.1.2 Thực nghiệm số 2

Với thực nghiệm số 2, em sẽ tiến hành đo khi có 2 đèn truyền dữ liệu và khoảng cách các tâm bóng đèn cách nhau 25 cm. Ngoài ra, vẫn có trường hợp thêm 1 bóng đèn sáng liên tục như trong thực nghiệm số 1 làm nhiễu.

a. **Trường hợp 2.1**: cả 2 đèn đều truyền dữ liệu với chuỗi 10 bits "011100 1001" (4 bits dữ liệu: 1001) ở tần số 1000Hz, ở vị trí trục nối tâm 2 đèn LED vuông góc với camera điện thoại như hình dưới đây:



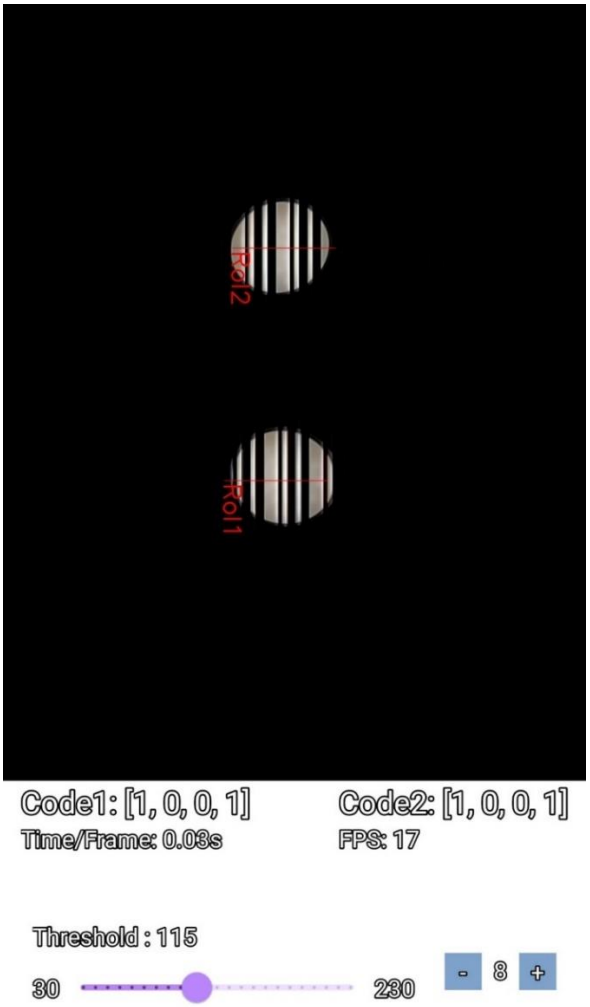
Hình 3. 3 Thực nghiệm trường hợp 2.1

Từ đó ta thu được số liệu trong Bảng 3. 3:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 2 LED (%)	
	LED 1	LED 2
30	87	84
40	92	94
50	96	99
60	95	98
70	90	92
80	66	80

Bảng 3. 3 Giá trị thực nghiệm trường hợp 2.1

b. Trường hợp 2.2: cả 2 đèn đều truyền dữ liệu vẫn với chuỗi 10 bits "011100 1001" ở tần số 1000Hz, ở vị trí trục nối tâm 2 đèn LED song song với camera điện thoại như hình dưới đây:



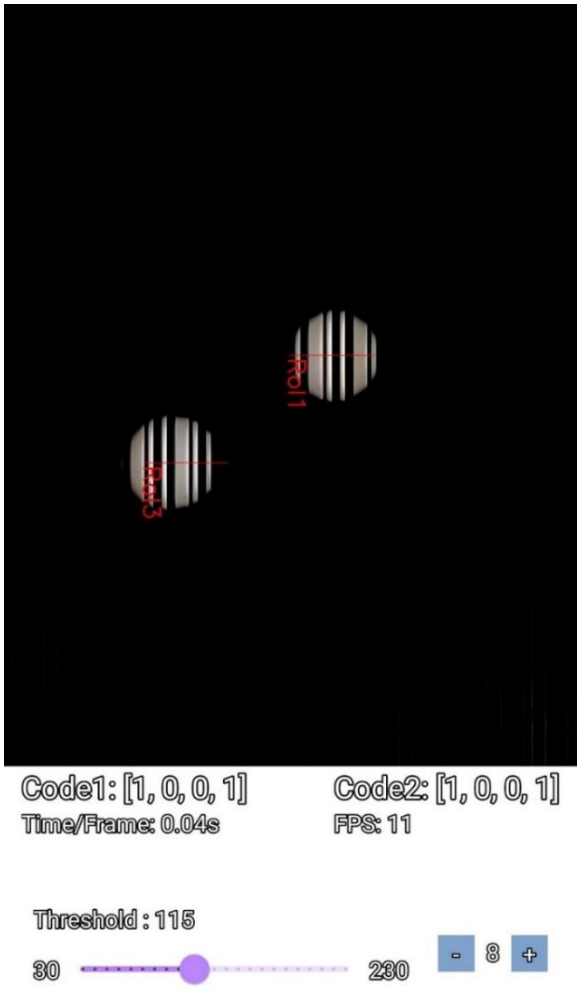
Hình 3. 4 Thực nghiệm trường hợp 2.2

Từ đó ta thu được số liệu trong Bảng 3. 4:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 2 LED (%)	
	LED 1	LED 2
30	91	91
40	93	91
50	99	97
60	98	96
70	85	80
80	70	67

Bảng 3. 4 Giá trị thực nghiệm trường hợp 2.2

c. Trường hợp 2.3: cả 2 đèn đều truyền dữ liệu vẫn với chuỗi 10 bits "011100 1001" ở tần số 1000Hz, ở vị trí trục nối tâm 2 đèn LED có góc chéo $\alpha = 45°$ so với camera điện thoại như hình dưới đây:



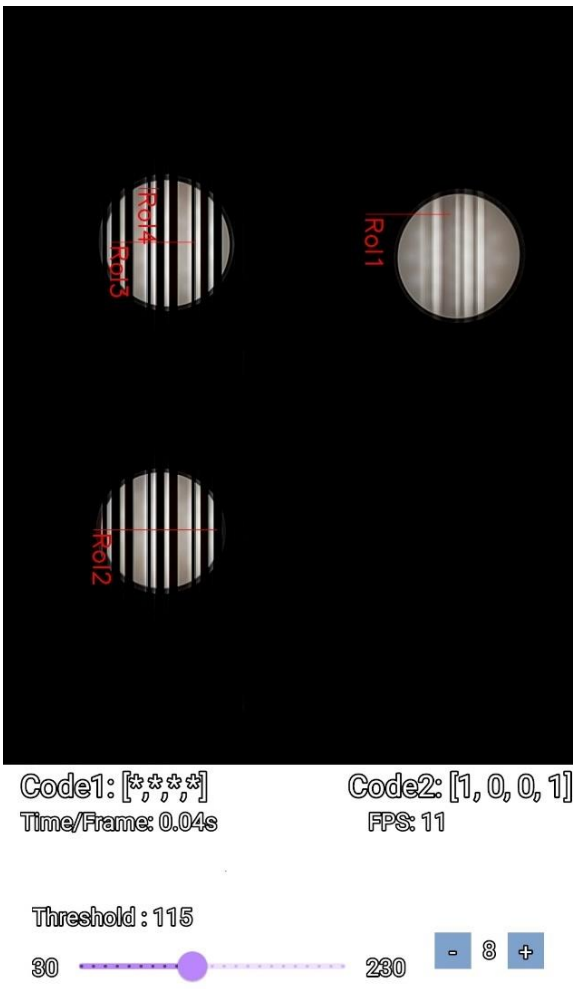
Hình 3. 5 Thực nghiệm trường hợp 2.3

Từ đó ta thu được số liệu trong Bảng 3. 5:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 2 LED (%)	
	LED 1	LED 2
30	85	86
40	87	89
50	97	99
60	93	95
70	89	90
80	80	85
90	60	56

Bảng 3. 5 Giá trị thực nghiệm trường hợp 2.3

d. Trường hợp 2.4: có 3 đèn LED chiếu sáng, trong đó có 2 đèn LED truyền dữ liệu vẫn với chuỗi 10 bits "011100 1001" ở tần số 1000Hz và 1 đèn chiếu sáng liên tục như hình dưới đây:



Hình 3. 6 Thực nghiệm trường hợp 2.4

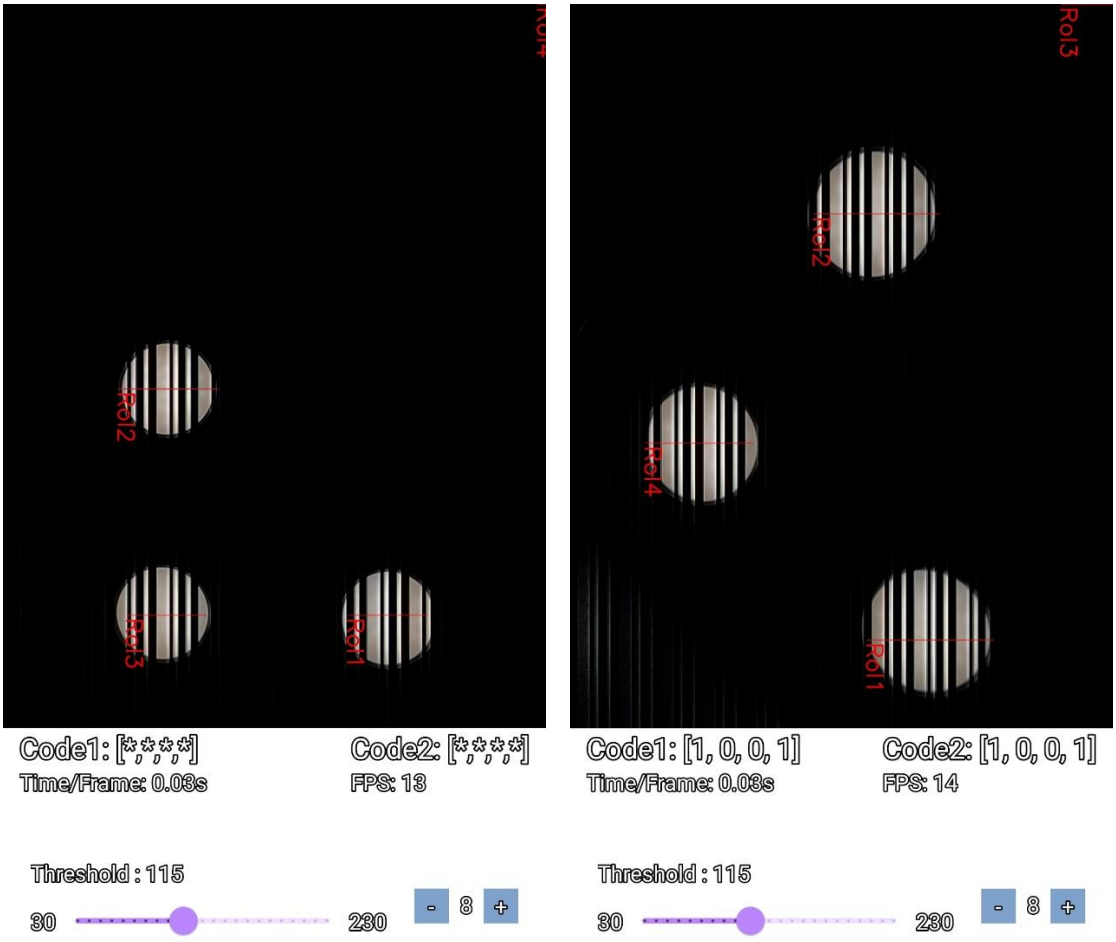
Từ đó ta thu được số liệu trong Bảng 3. 6:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 2 LED (%)	
	LED 1	LED 2
30	81	82
40	88	91
50	93	96
60	91	95
70	85	90
80	66	75

Bảng 3. 6 Giá trị thực nghiệm trường hợp 2.4

3.1.3 Thực nghiệm số 3

Có cả 3 đèn LED chiếu sáng và truyền dữ liệu vẫn với chuỗi 10 bits "011100 1001" ở tần số 1000Hz ở các vị trí khác nhau như hình dưới đây:



Hình 3. 7 Thực nghiệm trường hợp 3

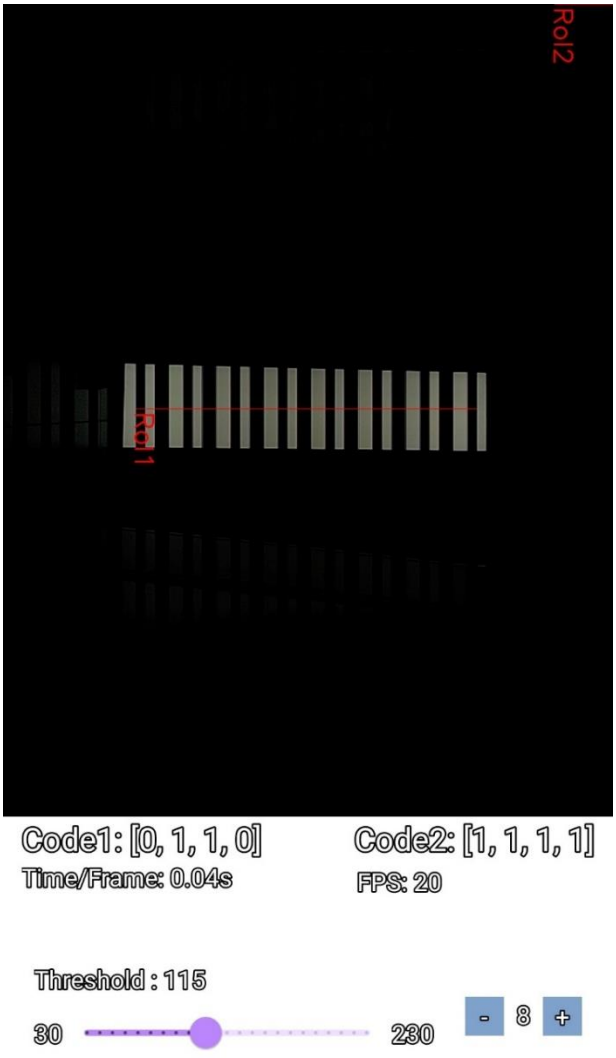
Từ đó ta thu được số liệu trong Bảng 3. 7:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác 3 LED (%)		
	LED 1	LED 2	LED 3
40	65	86	90
50	80	95	98
60	83	95	100
70	75	86	90
80	60	72	76

Bảng 3. 7 Giá trị thực nghiệm số 3

3.1.4 Thực nghiệm số 4

a. Trường hợp 4.1: đèn LED panel chiếu sáng và truyền dữ liệu với chuỗi 10 bit "011100 0110" (4 bits dữ liệu: 0110) ở tần số 1000Hz ở vị LED vuông góc với camera điện thoại như hình dưới đây:



Hình 3. 8 Thực nghiệm trường hợp 4.1

Từ đó ta thu được số liệu trong Bảng 3. 8:

Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác LED panel - 4 bits (%)
150	84
170	84
190	86
210	95
230	99
250	97
270	92
290	88
310	77

Bảng 3. 8 Giá trị thực nghiệm trường hợp 4.1

b. **Trường hợp 4.2:** đèn LED panel chiếu sáng và truyền dữ liệu với chuỗi 14 bits "011100 01101001" (8 bits dữ liệu: 0110 1001) ở tần số 1000Hz ở vị LED vuông góc với camera điện thoại như hình dưới đây:

Hình 3. 9 Thực nghiệm trường hợp 4.2

Từ đó ta thu được số liệu trong Bảng 3. 9:

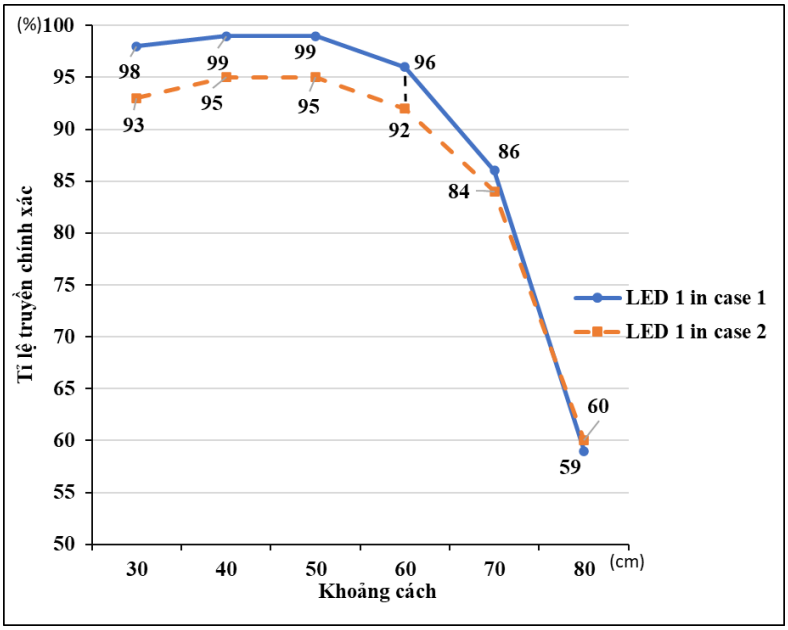
Khoảng cách giữa điện thoại và LED (cm)	Tỉ lệ truyền chính xác LED panel - 8 bits (%)
150	80
170	86
190	88
210	88
230	85
250	80
270	78
290	72
310	67

Bảng 3. 9 Giá trị thực nghiệm trường hợp 4.2

3.2 Nhận xét kết quả thực nghiệm

3.2.1 Thực nghiệm số 1

Đầu tiên, em sẽ đưa dữ liệu của thực nghiệm số 1.1 và 1.2 vào một biểu đồ đường để đánh giá sự khác biệt giữa việc có thêm đèn chiếu gây nhiễu và không có đèn nhiễu ảnh hưởng tới việc truyền nhận dữ liệu.



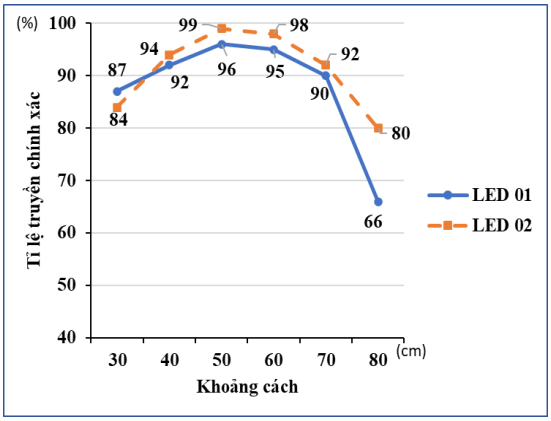
Hình 3. 10 Đồ thị đường trong thực nghiệm số 1

Từ Hình 3. 10, dễ thấy có sự tuyến tính trong mối quan hệ giữa khoảng cách và tỉ lệ chính xác khi truyền nhận. Trong khoảng cách từ 30cm đến 70cm tỉ lệ chính xác rất cao từ 95% đến 99% với cả 2 trường hợp. Từ khoảng cách 70cm trở đi thì

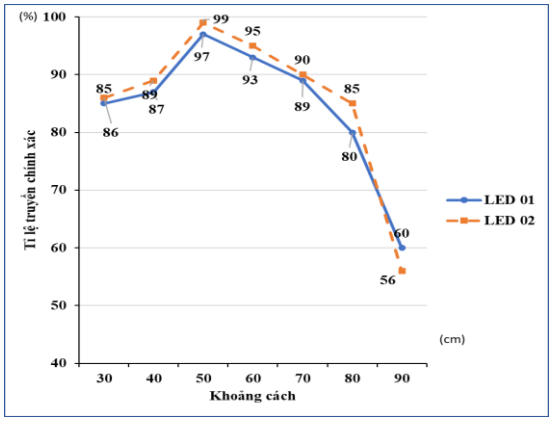
khoảng cách càng lớn, sai số càng nhiều. Do khoảng cách càng lớn hình ảnh đèn LED bé đi dẫn tới số sọc sáng (tối) giảm nghĩa là tỉ lệ nhận đủ mỗi frame truyền (10 bit/frame) bị giảm. Việc có thêm đèn nhiễu thì tỉ lệ truyền nhận chính xác có bị ảnh hưởng nhưng chênh lệch lớn nhất giữa 2 thực nghiệm chỉ ở mức $\Delta = 4\%$.

3.2.2 Thực nghiệm số 2

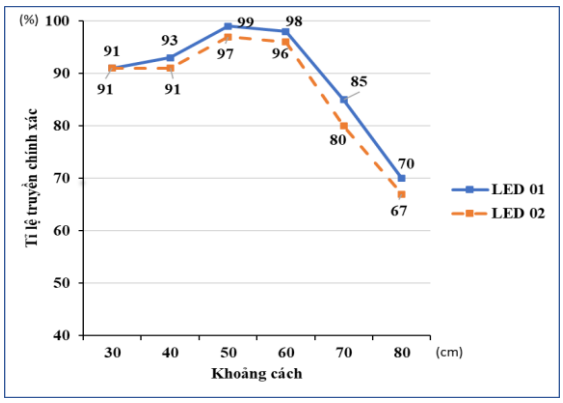
Tiếp theo, ta sang đánh giá về thực nghiệm với 2 đèn LED truyền dữ liệu trong 4 trường hợp qua 4 biểu đồ đường như sau:



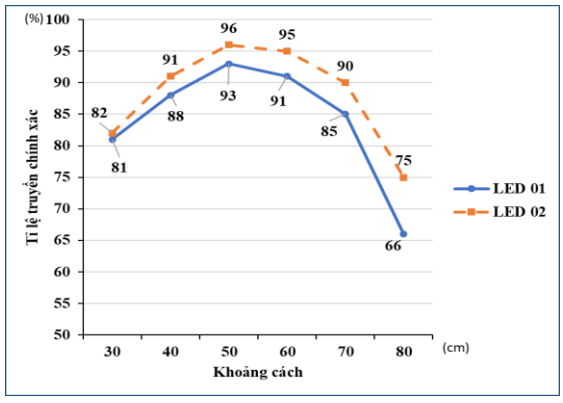
Thí nghiệm 2.1



Thí nghiệm 2.3



Thí nghiệm 2.2



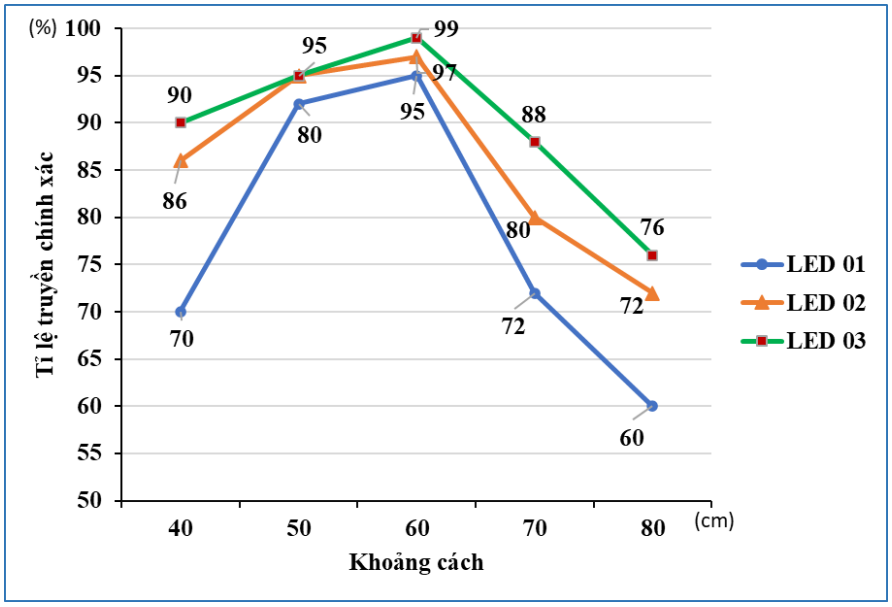
Thí nghiệm 2.4

Hình 3. 11 Đồ thị đường trong thực nghiệm số 2

Từ Hình 3. 11, ta nhận thấy cả 4 trường hợp vẫn có sự tuyến tính trong mối quan hệ giữa khoảng cách và tỉ lệ chính xác khi truyền nhận. Trong trường hợp 2.1, 2.2 và 2.3, tỉ lệ truyền nhận chính xác của 2 đèn LED chênh lệch không đáng kể dù camera tạo góc với cụm đèn LED ở các vị trí khác nhau. Trường hợp 2.4, khi có thêm đèn nhiễu chiếu cùng, tỉ lệ truyền nhận chính xác của mỗi đèn có giảm rất nhỏ khoảng 2-3% ở các vị trí khác nhau. Vì vậy, việc hai đèn phát tín hiệu trong môi trường đèn sáng bình thường xung quanh sẽ ảnh hưởng ít tới việc truyền dữ liệu.

3.2.3 Thực nghiệm số 3

Trường hợp 3 đèn LED cùng truyền dữ liệu, tương tự ta cũng có biểu đồ đường đánh giá ti lệ truyền nhận chính xác như sau:

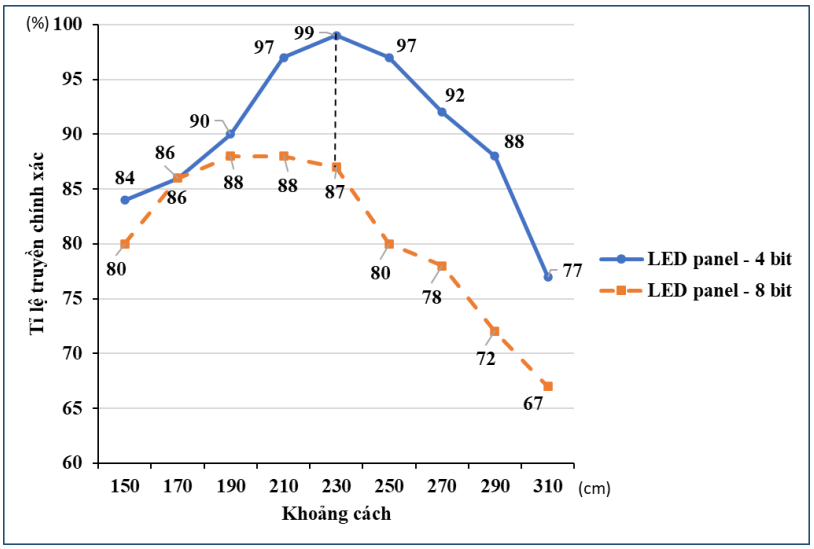


Hình 3. 12 Đồ thị đường trong thực nghiệm số 3

Nhận xét chung, xu hướng trong biểu đồ này tương tự như các biểu đồ trước đó, khi khoảng cách để đạt tỉ lệ chính xác cao nhất (95% – 99%) trong khoảng từ 50 – 60 cm. Tại trường hợp với 3 đèn này, khi ở vị trí quá gần, 1 trong 3 đèn sẽ có thể không được lấy toàn bộ đèn bởi camera dẫn đến đường RoI sẽ không đủ dài để có đủ frame truyền. Do đó tỉ lệ truyền chính xác sẽ thấp hơn 2 đèn còn lại tương đối nhiều.

3.2.4 Thực nghiệm số 4

Với thực nghiệm số 4, ta có biểu đồ đường như sau:



Hình 3. 13 Đồ thị đường trong thực nghiệm số 4

Khi thực nghiệm với đèn LED panel, xu hướng của biểu đồ cũng tương tự với các biểu đồ trước đó nhưng ta thấy khoảng cách để tỉ lệ truyền nhận dữ liệu chính xác

cao đã gia tăng đáng kể. Cụ thể tỉ lệ chính xác khi truyền 4 bits dữ liệu lên tới 99% ở khoảng cách 230cm. Đây chính là khoảng cách sử dụng khi thiết kế hệ thống đèn, nhằm đạt tỉ lệ lớn nhất khi truyền với khoảng cách 2,1m – 2,5m, do độ cao trần trong khoảng 3m. Ngoài ra với thực nghiệm truyền dữ liệu 8 bit, tỉ lệ chính xác đã bị giảm xuống khoảng 10% cùng với khoảng cách như vậy. Lí do giảm là mảng giá trị 0 và 1 sau khi dùng curve fit, có những điểm lập lờ có giá trị rất gần đường curve fit, đó là những giá trị ảnh hưởng đến việc giải mã sai tăng lên.

CHƯƠNG 4. KẾT LUẬN

4.1 Kết luận

Sau quá trình làm đồ án tốt nghiệp với đề tài "Thiết kế phần mềm trên thiết bị di động sử dụng truyền thông bằng ánh sáng nhìn thấy được (VLC) cho ứng dụng định vị trong nhà" em đã có được những kiến thức về:

- Tìm hiểu tổng quan về công nghệ VLC được áp dụng cho ứng dụng định vị trong nhà
- Lập trình phần mềm trên thiết bị di động chạy hệ điều hành Android với ngôn ngữ Kotlin
- Lập trình Python, thị giác máy trên điện thoại di động

Sản phẩm đồ án của em thu được kết quả như sau:

- Giải mã dữ liệu được truyền từ nhiều đèn LED cùng một thời điểm
- Phần mềm có thể chạy trên nhiều dòng điện thoại khác có cùng hệ điều hành Android
- Phần mềm viết bằng ngôn ngữ Kotlin có tích hợp Python để xử lý ảnh có hiển thị RoI, giải mã dữ liệu và tính toán tỉ lệ chính xác.
- Xác định mối quan hệ khoảng cách giữa máy ảnh và đèn LED, số lượng đèn, tần số truyền tin đến chất lượng truyền tin.

4.2 Hướng phát triển

Với sự phát triển ngày càng mạnh mẽ của công nghệ máy ảnh trên điện thoại di động và công nghệ đèn LED, đồ án này có nhiều tiềm năng để được cải tiến và phát triển. Các hướng cải thiện bao gồm:

- Tối ưu hóa thuật toán tìm ngưỡng động để đạt kết quả tốt hơn
- Tối ưu hóa thuật toán tìm RoI cho nhiều đèn
- Cải thiện thời gian xử lý với từng ảnh, đồng thời cải thiện phương pháp tính tỉ lệ chính xác sao cho tối ưu nhất cho việc ứng dụng.
- Xây dựng thuật toán để khi đã có dữ liệu từng đèn rồi xác định vị trí của người dùng.

Định vị trong nhà đang là một nhu cầu cần thiết ở hiện tại và tương lai. Việc áp dụng VLC vào định vị trong nhà kết hợp với công nghệ 5G/6G trong tương lai sẽ là một giải pháp mà có thể giải quyết rất nhiều vấn đề còn đang gặp phải.

TÀI LIỆU THAM KHẢO

[1] K. L. Ullah, "Visible light communication: Applications, architecture, standardization and research challenges," 2017.

[2] C. Ley-Bosch, I. Alonso-González, D. Sasnchez-Rodríguez, C. Ramíez-Casanas, "Evalution of the effects of hidden node problems in IEEE 802.15.7 uplink perfomance," *Sensor 16*, 2016.

[3] IEEE, *P802.15.7 – Standard for Short-Range Wireless Optical Communication*, 2011.

[4] Y. Wang, Y. Wang, N. Chi, J. Yu, H. Shang, "Demonstration of 575-Mb/s downlink and 225-Mb/s uplink bi-directional SCM-WDM visible light communication using RGB LED and phosphor-based LED," *Opt. Express 21 (2013) 1203–1208.*.

[5] M. Craford, "Visible light emitting diode technology: High performance, more colors, and moving into incandescent lamp applications," *in: Proceedings of the Quantum Electronics and Laser Science Conference*, 1996..

[6] D. Wu, Z. Ghassemlooy, H. Le Minh, S. Rajbhandari, M. A Khalighi, and X. Tang, "Optimisation of Lambertian order for indoor non-directed optical wireless communication," *IEEE Xplore,* 2012.

[7] E.F. Schubert, T. Gessmann, J.K Kim, "Light Emitting Diodes," *Wiley Online Library*, 2005.

[8] K. Sindhubala, B. Vijayalakshmi, "Design and implementatin of visible ligh communication system in indoor environment," *ARPN, J. Eng. Appl. Sci (7)*, 2006.

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

[10] L.Takai, S. Ito, K. Yasutomi, K. Kagawa, M Andoh, and S. Kawahito, "LED and CMOS Image Sensor Based Optical Wireless Communication System for Automotive Applications," *IEEE Photonics Journal*, pp. vol. 5, no. 5, pp. 6801418, 2013.

[11] Gamal, A.E; Eltoukhy, "H. CMOS Image Sensors," *IEEE Circuits Devices Mag*, 2005.

[12] Chen, Z.; Wang, X.; Pacheco, S; Liang, , "R. Impact of CCD camera SNR on polarimetric accuracy," *Appl. Opt*, pp. 53, 7649-7656, 2014.

[13] S. Electronics, "Tutorial on Visible Light Communication," *IEEE 802.15.*

[14] C. V. S. C. Consortium, "Vehicle safety communications project: Task 3 final report: identify intelligent vehicle safety applications enabled by DSRC," *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC,* 2005.

[15] Ruei-Jie Shiu, Yen-Chun Liu, Chi-Wai Chow, "Visible Light Communication Using Advertisement-Light-Board and Rolling-Shutter-Effect based CMOS Mobile-Phone Camera," *The 23rd Opto Electronics and Communications Conference* , July 02-06, 2018.

[16] Anurag Sarkar, Shalabh Agarwal, Asoke Nath, "Li-Fi technology: data transmission through visbile light," *Int. J. Adv. Res. Comput. Sci. Manag. Stud. 3 (6),* 2015.

[17] H. Nguyen, M. D. Thieu, T. L. Pham, H. Nguyen, and Y. M. Jang, "The Impact of Camera Parameters on Optical Camera Communication," *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC),* Feb. 2019.

[18] R. McEliece, "The Theory of Information and Coding," *Cambridge University Press, Cambridge, United Kingdom,* 2002.

[19] C. Langton, "Tutorial 12 Coding and decoding with Convolutional Codes.," *Complex2Real.com Complex Communications, Technol,* Made Easy (1999).

[20] N. Chi, Yuanquan Wang, Yiguang Wang, X. Huang, X. Lu, "Ultra-high-speed single red-green-blue light-emitting diode-based visible light communication system utilizing advanced modulation formats," *Chin. Opt. Lett,* 12 (2014).

[21] W. Zhang, M. I. S. Chowdhury, and M. Kavehrad, "Asynchronous indoor positioning system based on visible light communication," *Optical Engineering,* Vols. 53, no.4, p. 045105, 2014.

PHỤ LỤC

A1. Chương trình khởi tạo camera, gọi hàm xử lý ảnh và kết quả trả về hiển thị lên Textureview

```
private val surfaceTextureListener = object :
TextureView.SurfaceTextureListener {
    override fun onSurfaceTextureAvailable(surface: SurfaceTexture,
width: Int, height: Int) {
        openCamera(width, height)
        startFPSCounter()
    }

    override fun onSurfaceTextureSizeChanged(surface: SurfaceTexture,
width: Int, height: Int) {
        configureTransform(width, height)
    }

    override fun onSurfaceTextureDestroyed(surface: SurfaceTexture):
Boolean {
        stopFPSCounter()
        return true
    }

    override fun onSurfaceTextureUpdated(surface: SurfaceTexture) {
        updateVIew() {
            isFirst = false
        }
        binding.apply {
            val bitmap = cameraView.bitmap
            bitmap?.let {
                val size: Int = it.rowBytes * it.height
                val byteBuffer = ByteBuffer.allocate(size)
                bitmap.copyPixelsToBuffer(byteBuffer)
                val byteArray: ByteArray = byteBuffer.array()
                val result = processFrame.callAttr(
                    "process_frame",
                    byteArray,
                    it.height,
                    it.width,
                    thresholdDefault,
                    nPixel
                )
                try {
                    val obj = result.toJava(Data::class.java)
                    if (obj.byte?.isEmpty() == false) {
                        val bitmaps =
BitmapFactory.decodeByteArray(obj.byte, 0, obj.byte.size)
                        viewImage.setImageBitmap(bitmaps)

                    }
                catch (_: Exception) {
                    }
```

A2. Chương trình tính giá trị ngưỡng và Npixel tự động

```
def process_frame(frame, heigth, width, thresholdDefault, Npixel):
    try:
        st = time.time()

        img = np.frombuffer(frame, dtype=np.uint8).reshape(heigth,
width, 4)
        img = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
        frame = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        blurred = cv2.GaussianBlur(frame, (7, 7), 0)
        select = frame[frame > 10]
        avg = np.mean(select)
        print(f"avg pixel value: {avg}")

        width = frame.shape[1]

        # ret, frame0 = cv2.threshold(blurred, 0, 255,
cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        ret, frame0 = cv2.threshold(blurred, avg, 255,
cv2.THRESH_BINARY)
        print(f'ret of otsu:{ret}')
        contours, hierarchy = cv2.findContours(frame0, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        print("len contours: ", len(contours))
        if len(contours) > 0 and len(contours)<=70:
            contours = contours
        if len(contours) > 70 and len(contours)<90:
            contours = contours[0:(len(contours)-20)]
        if len(contours) >= 90 and len(contours)<100:
            contours = contours[0:len(contours):2]
        if len(contours) >= 100 and len(contours)<150:
            contours = contours[0:len(contours):3]
        if len(contours) >= 150 and len(contours)<200:
            contours = contours[0:len(contours):4]
        if len(contours) >= 200 and len(contours)<300:
            contours = contours[0:len(contours):5]
        # print("lennnnnnn6",len(contours))

            print("contour over")
        print(f"len contours sau khi loc:{len(contours)} ")
        mass_centres_x = []
        mass_centres_y = []
        top = []
        bot = []
        height = []
        for i in range(0, len(contours)):
            M = cv2.moments(contours[i], 0)
            if M["m00"] != 0:
                mass_centres_x.append(int(M['m10']/M['m00']))
                mass_centres_y.append(int(M['m01']/M['m00']))
            else:
                mass_centres_x.append(int(0))
                mass_centres_y.append(int(0))

        for i in range(0, len(contours)):
            x,y,w,h = cv2.boundingRect(contours[i])
            cv2.rectangle(frame0,(x,y),(x+w,y+h),(255,0,0),2)
            top.append(int(y))
            bot.append(int(y+h))
            height.append(int(h))
```

```
        sort_height = sorted(height, reverse=True)
        Npixel = int(sort_height[0]/3)

        # Npixel =13
        print(f'sort_height:{sort_height[:]}')
        print(f'Npixel of sort: {Npixel}')

        if len(contours) == 0 or len(contours) > 80:
            pass
        else:
            if len(mass_centres_x) == 0:
                mass_centres_x = np.zeros(5, dtype=int)
            if len(mass_centres_y) == 0:
                mass_centres_y = np.zeros(5, dtype=int)
            if len(top) == 0:
                top = np.zeros(5, dtype=int)
            if len(bot) == 0:
                bot = np.zeros(5, dtype=int)
```