**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

A red and white logo

Description automatically generated

**ĐỒ ÁN TỐT NGHIỆP**

**NGHIÊN CỨU PHƯƠNG PHÁP MÃ HOÁ MỚI NHẰM NÂNG CAO KHẢ NĂNG TRUYỀN DỮ LIỆU CHO TRUYỀN THÔNG BẰNG ÁNH SÁNG KHẢ KIẾN**

**LÊ VĂN ĐỒNG**

dong.lv191749@sis.hust.edu.vn

**Ngành Kỹ thuật Điều khiển và Tự động hoá**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Nguyễn Hoàng Nam  Chữ ký của GVHD |
| **Khoa:** | Tự động hoá |
| **Trường :** | Điện – Điện tử |
| **HÀ NỘI, 01/2024** | |

|  |  |
| --- | --- |
| 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: Lê Văn Đồng

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

1. *Tên đề tài:* **Nghiên cứu phương pháp mã hoá mới nhằm nâng cao khả năng truyền dữ liệu cho truyền thông bằng ánh sáng khả kiến.**
2. *Nội dung đề tài:*
   1. *Các số liệu ban đầu:*

* Phần cứng (đã được thiết kế) gồm:
* Cụm đèn 4 đèn LED âm tường
* Camera trên điện thoại di động Aris Pro 8
* Phần mềm gồm:
* Android studio
* Visual Studio Code
* Ngôn ngữ Python, Kotlin
  1. *Các nội dung tính toán, thiết kế:*
* Nghiên cứu, tìm hiểu về công nghệ giao tiếp không dây quang học (OWC)
* Ứng dụng giao tiếp camera quang học OCC 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.
* Đề xuất phương pháp điều chế dữ liệu mới cải thiện hiệu suất truyền tin
* Thiết kế phần mềm phần phát:
* Truyền dữ liệu với tần số và dữ liệu thay đổi được
* Quản lí dữ liệu trên OTA Drive
* Thiết kế phần mềm phần thu:
* Sử dụng ngôn ngữ Python, Kotlin và các thư viện xử lý ảnh như OpenCV, Numpy,… để phát triển phần mềm trên thiết bị di động.
* Áp dụng ngưỡng chuyển đổi ảnh thành ảnh nhị phân
* Thuật toán tìm RoI và đọc dữ liệu từ ảnh thu được
* Giải mã và trả về thiết bị
* Tính toán tỉ lệ truyền chính xác trong quá trình truyền.

1. *Thời gian giao đề tài:* 10/2023
2. *Thời gian hoàn thành:* 01/2024

*Ngày 21 tháng 01 năm 2024*

**CÁN BỘ HƯỚNG DẪN**

**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 - giảng viên 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 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 thuộc trường Điện – Điện Tử và ban giám hiệu Đại Học Bách Khoa Hà Nội 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 và kiến thức còn hạn chế nên nghiên cứu này có thể chưa đáp ứng trọn vẹn nhất 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 tổng quan về công nghệ giao tiếp không dây quang học (OWC) đặc biệt là giao tiếp camera quang học (OCC) ứng dụng định vị trong nhà
* Các thông số kĩ thuật ảnh hưởng đến quá trình truyền và giải mã dữ liệu
* Lập trình phần mềm trên thiết bị di động chạy hệ điều hành Android
* Đề xuất phương pháp điều chế cải thiện hiệu suất truyền
* Phát triển thuật toán xử lí ảnh tăng hiệu quả đọc dữ liệu và tốc độ xử lí
* Đo tỉ lệ truyền nhận chính xác, so sánh hiệu quả đạt được so với các hệ thống trước đó

Công cụ, phần mềm sử dụng:

* Cụm LED trần 4 bóng
* Module ESP32-WROVER
* Phần mềm Android studio, Visual Studio Code. Ngôn ngữ Python, Kotlin,C/C++

Kết quả:

* Lập trình cho Chip điều khiển ESP32 Wrover với nhiệm vụ: kiểm soát tần số nháy LED theo dữ liệu vị trí, quản lí dữ liệu truyền trên OTA driver
* Triển khai thành công hệ thống điều chế MPPM giúp tăng kích thước dữ liệu truyền vẫn đảm bảo không xảy ra hiện tượng nhấp nháy cho mắt người.
* Tối ưu hoá phần mềm cung cấp hình ảnh và trả kết quả về giao diện người dùng chạy trên hệ điều hành Android
* Đề xuất thuật toán xử lí ảnh giúp phát hiện RoI và tăng tốc đọc dữ liệu từ hình ảnh thu được với độ chính xác cao
* Giải mã giữ liệu và trả về dữ liệu từng đèn LED

Định hướng phát triển:

* Tối ưu hệ thống phát giúp quản lí dữ liệu linh hoạt hơn
* Tối ưu hoá phần mềm, tăng tốc độ xử lí dữ liệu tiếp cận hơn với người sử dụng
* Xác định nhân tố chính để tăng khoảng cách truyền nhận giữa đèn LED và camera.
* Xây dựng thuật toán định vị từ dữ liệu thu được của từng đèn LED

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

A close up of a word

Description automatically generated

**Lê Văn Đồng**

**MỤC LỤC**

[CHƯƠNG 1. TỔNG QUAN TRUYỀN THÔNG CAMERA QUANG HỌC 1](#_Toc158963417)

[1.1 Giới thiệu về công nghệ giao tiếp không dây quang học (OWC) 1](#_Toc158963418)

[1.2 Tổng quan về hệ thống OCC dựa trên đèn LED 2](#_Toc158963419)

[1.2.1 Nguyên tắc hoạt động hệ thống OCC 2](#_Toc158963420)

[1.2.2 Bộ phát 3](#_Toc158963421)

[1.2.3 Đầu thu 4](#_Toc158963422)

[1.3 Các ứng dụng tiềm năng 6](#_Toc158963423)

[1.3.1 Hệ thống giao thông thông minh dựa trên OCC 6](#_Toc158963424)

[1.3.2 Kết nối IoT dựa trên OCC 7](#_Toc158963425)

[1.3.1 Định vị trong nhà (IPS) 8](#_Toc158963426)

[CHƯƠNG 2. TỔNG QUAN KĨ THUẬT SỬ DỤNG TRONG HỆ THỐNG OCC 11](#_Toc158963427)

[2.1 Các kĩ thuật điều chế cho hệ thống LED- Camera 11](#_Toc158963428)

[2.1.1 Kĩ thuật điều chế On Off Keying (OOK) 11](#_Toc158963429)

[2.2 Kĩ thuật điều chế MPPM 13](#_Toc158963430)

[2.2.1 Giới thiệu 13](#_Toc158963431)

[2.2.2 Bộ mã hoá 15](#_Toc158963432)

[2.2.3 Bộ giải mã 17](#_Toc158963433)

[2.3 Thị giác máy trên thiết bị di động. 17](#_Toc158963434)

[2.3.1 Tổng quan về ảnh kỹ thuật số 17](#_Toc158963435)

[2.3.2 Thị giác máy trong VLC 21](#_Toc158963436)

[2.4 Các thông số ảnh hưởng đến hiệu suất truyền tin 23](#_Toc158963437)

[2.4.1 Tốc độ khung hình 23](#_Toc158963438)

[2.4.2 Kích thước LED trong ảnh 26](#_Toc158963439)

[2.4.3 Cấu trúc khung bản tin 27](#_Toc158963440)

[CHƯƠNG 3. THIẾT KẾ HỆ THỐNG 30](#_Toc158963441)

[3.1 Phần phát (Tx) 30](#_Toc158963442)

[3.1.1 Đèn LED 31](#_Toc158963443)

[3.1.2 ESP32 WROVER 31](#_Toc158963444)

[3.1.3 Lập trình phần phát 33](#_Toc158963445)

[3.1.4 Kết quả triển khai phần phát 39](#_Toc158963446)

[3.2 Phần thu (Rx) 39](#_Toc158963447)

[3.2.1 Thiết bị sử dụng cho phần thu 40](#_Toc158963448)

[3.2.2 Thuật toán xử lí ảnh 40](#_Toc158963449)

[CHƯƠNG 4. KẾT QUẢ THỰC NGHIỆM 45](#_Toc158963450)

[4.1 Triển khai thực nghiệm 45](#_Toc158963451)

[4.1.1 Thực nghiệm với tần số 14kHz 45](#_Toc158963452)

[4.1.2 Thực nghiệm với tần số 18kHZ 46](#_Toc158963453)

[4.1.3 Thực nghiệm với tần số 20kHz 47](#_Toc158963454)

[4.2 Đánh giá kết quả đạt được 48](#_Toc158963455)

[4.2.1 Tần số 14kHz 48](#_Toc158963456)

[4.2.2 Tần số 18kHz 49](#_Toc158963457)

[4.2.3 Tần số 20kHz 50](#_Toc158963458)

[CHƯƠNG 5. KẾT LUẬN 51](#_Toc158963459)

[5.1 Kết luận 51](#_Toc158963460)

[5.2 Hướng phát triển 51](#_Toc158963461)

[TÀI LIỆU THAM KHẢO 52](#_Toc158963462)

[PHỤ LỤC 55](#_Toc158963463)

**DANH MỤC HÌNH VẼ**

[Hình 1‑1: Phân loại hệ thống OWC 1](#_Toc158882292)

[Hình 1‑2: Sơ đồ khối hệ thống OCC 2](#_Toc158882293)

[Hình 1‑3: Các loại đèn LED trắng 4](#_Toc158882294)

[Hình 1‑4: Sự khác biệt giữa cảm biến CCD và cảm biến CMOS 5](#_Toc158882295)

[Hình 1‑5: Cảm biến CMOS quét đèn LED ở trạng thái ON và OFF 6](#_Toc158882296)

[Hình 1‑6: Hệ thống giao thông thông minh dựa trên OCC 7](#_Toc158882297)

[Hình 1‑7: Nông nghiệp thông minh dựa trên OCC – IoT 8](#_Toc158882298)

[Hình 1‑8: Nhu cầu sử dụng đèn LED đến năm 2030 8](#_Toc158882299)

[Hình 1‑9: Hệ thống định vị trong nhà dựa trên OCC 10](#_Toc158882300)

[Hình 2‑1: On-off keying (OOK bằng đèn LED 12](#_Toc158882301)

[Hình 2‑2: Loại bỏ hiện tượng nhấp nháy bằng mã hoá Manchester 12](#_Toc158882302)

[Hình 2‑3: Cấu trúc khung dữ liệu: (a) Khung dữ liệu (b) Khung truyền có mã hóa manchester 12](#_Toc158882303)

[Hình 2‑4: Các khung hình được chụp của điều chế OOK với các tần số khác nhau 13](#_Toc158882304)

[Hình 2‑5: Khối tín hiệu trong MPPM 14](#_Toc158882305)

[Hình 2‑6: Toạ độ điểm ảnh và màu sắc 18](#_Toc158882306)

[Hình 2‑7: Ảnh màu 32 bits 19](#_Toc158882307)

[Hình 2‑8: Ảnh xám 8 bit 19](#_Toc158882308)

[Hình 2‑9: Ảnh nhị phân áp dụng ngưỡng 20](#_Toc158882309)

[Hình 2‑10: Ảnh nhị phân với từng giá trị ngưỡng khác nhau 20](#_Toc158882310)

[Hình 2‑11: Nhận diện khuôn mặt trên iPhone X bằng công nghệ Face ID 21](#_Toc158882311)

[Hình 2‑12: Dải sáng/tối của hệ thống OCC 24](#_Toc158882312)

[Hình 2‑13: Quá trình xử lý các khung liền kề của cảm biến CMOS a) Trường hợp lí tưởng b)Trường hợp thực tế 24](#_Toc158882313)

[Hình 2‑14: Thời gian cho mỗi khung hình 25](#_Toc158882314)

[Hình 2‑15: Mỗi liên hệ giữ hình ảnh thu được trên cảm biến và hình ảnh thực tế 26](#_Toc158882315)

[Hình 2‑16: Ảnh chụp cụm 4 đèn Led ở khoảng cách 1.7m và kích thước của chúng 27](#_Toc158882316)

[Hình 2‑17: Thông số camera của 1 thiết bị di động 28](#_Toc158882317)

[Hình 3‑1: Sơ đồ phần cứng Tx 30](#_Toc158882318)

[Hình 3‑2: Sơ đồ nguyên lí Tx 30](#_Toc158882319)

[Hình 3‑3: Đèn Led âm trần 31](#_Toc158882320)

[Hình 3‑4: Chip điều khiển ESP32 Wrover 32](#_Toc158882321)

[Hình 3‑5: Hình ảnh thực tế hệ thống Tx 33](#_Toc158882322)

[Hình 3‑6: Lưu đồ mã hoá dữ liệu và điều khiển nháy LED 34](#_Toc158882323)

[Hình 3‑7: Nguyên lí vận hành kết nối không dây cho ESP32 36](#_Toc158882324)

[Hình 3‑8: Đặt tên sản phẩm và version mặc định 36](#_Toc158882325)

[Hình 3‑9: Copy API key 37](#_Toc158882326)

[Hình 3‑10: Mở tab firmware và ấn add new 38](#_Toc158882327)

[Hình 3‑11: Upload các phiên bản của bạn 38](#_Toc158882328)

[Hình 3‑12: Chọn phiên bản cần sử dụng, sau đó nhấn Save 38](#_Toc158882329)

[Hình 3‑13: Hình vẽ Frame truyền chi tiết 39](#_Toc158882330)

[Hình 3‑14: Biểu đồ tuần tự của phần mềm 39](#_Toc158882331)

[Hình 3‑15: Sơ đồ khối quá trình xử lí ảnh 40](#_Toc158882332)

[Hình 3‑16a: Ảnh màu 41](#_Toc158882333)

[Hình 3‑17b: Ảnh xám 41](#_Toc158882334)

[Hình 3‑18: Ảnh nhị phân đã vẽ contours 42](#_Toc158882335)

[Hình 3‑19: Lưu đồ thuật toán đọc dữ liệu trên LED 43](#_Toc158882336)

[Hình 3‑20: Toạ độ các bounding box 44](#_Toc158882337)

[Hình 4‑1a: Kết quả chạy trên app Android 45](#_Toc158882338)

[Hình 4‑2b: Thông số thiết bị 1 (camera trước) 45](#_Toc158882339)

[Hình 4‑3: Đồ thị đường trong thực nghiệm với tần số 14kHz 48](#_Toc158882340)

[Hình 4‑4: Đồ thị đường trong thực nghiệm với tần số 18kHz 49](#_Toc158882341)

[Hình 4‑5: Đồ thị đường trong thực nghiệm với tần số 20kHz 50](#_Toc158882342)

**DANH MỤC BẢNG BIỂU**

[Bảng 1‑1: So sánh hệ thống OCC và VLC 2](#_Toc158882343)

[Bảng 1‑2: So sánh các công nghệ IPS hiện có 9](#_Toc158882344)

[Bảng 2‑1: Các phương pháp điều chế đề xuất 11](#_Toc158882345)

[Bảng 2‑2: Ví dụ cho điều chế MPPM với n=8 14](#_Toc158882346)

[Bảng 2‑3: Ma trận mã hóa và giải mã ký hiệu 15](#_Toc158882347)

[Bảng 2‑4: Ví dụ minh họa mã hóa kí tự 15](#_Toc158882348)

[Bảng 2‑5: Chuỗi mã hóa cho n = 5 16](#_Toc158882349)

[Bảng 2‑6: Biểu đồ màu 18](#_Toc158882350)

[Bảng 2‑7: Cấu trúc khung bản tin 29](#_Toc158882351)

[Bảng 3‑1: Thông số đèn LED 31](#_Toc158882352)

[Bảng 3‑2: Thông số phần cứng ESP32. 32](#_Toc158882353)

[Bảng 3‑3: Thông số phần mềm ESP32 32](#_Toc158882354)

[Bảng 4‑1: Kết quả thực nghiệm với tần số 14kHZ 45](#_Toc158882355)

[Bảng 4‑2: Kết quả thực nghiệm với tần số 18kHZ 46](#_Toc158882356)

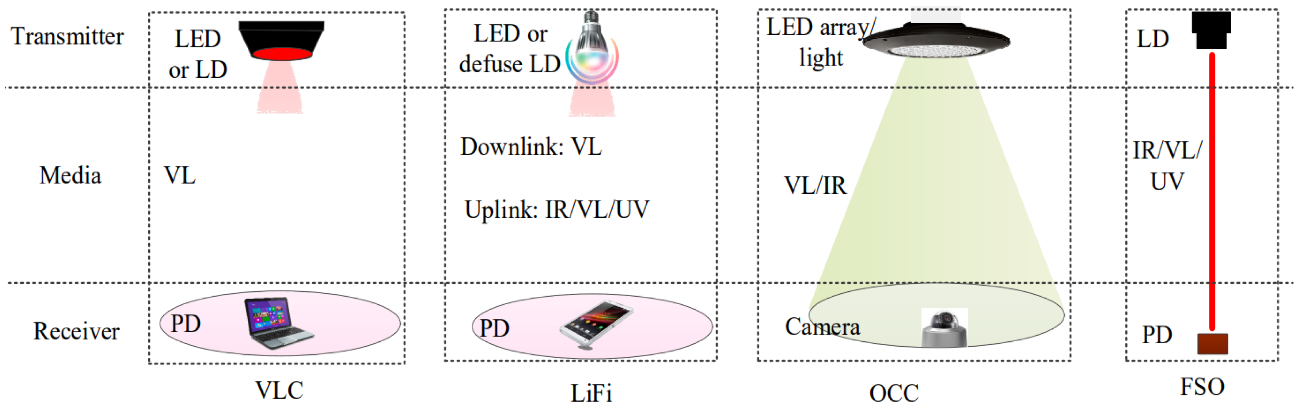
[Bảng 4‑3: Kết quả thực nghiệm với tần số 20kHZ 47](#_Toc158882357)

# TỔNG QUAN TRUYỀN THÔNG CAMERA QUANG HỌC

## Giới thiệu về công nghệ giao tiếp không dây quang học (OWC)

Với sự phát triển chưa từng có và đột phá của công nghệ thiết bị không dây, các chức năng của truyền thông không dây đã được mở rộng đáng kể. Tuy nhiên, việc cung cấp dịch vụ đa phương tiện chất lượng cao cho người dùng đòi hỏi một lượng băng thông đáng kể và phủ sóng mạng rộng, điều này đồng nghĩa với việc khả năng dung lượng phổ tần hiện tại đang trải qua "sự khan hiếm về phổ" do sự hạn chế về sẵn có [1]. Để giảm nhẹ tình trạng quá tải phổ tần radio frequency (RF), các nhà nghiên cứu đang khám phá việc sử dụng các dải tần số cao như sóng millimet [2] [3], terahertz ,và băng petahertz.

Là một giải pháp bổ sung hấp dẫn, giao tiếp không dây quang học (OWC) đã được xem xét để giảm áp lực sử dụng băng thông trên hệ thống không dây RF, vì nó bao gồm toàn bộ phổ quang từ 350 đến 1550 nm. Khác với băng thông 300 GHz trong công nghệ RF, các công nghệ OWC cung cấp một băng thông lý thuyết lớn lên đến 20 THz cho hồng ngoại (IR), 320 THz cho ánh sáng nhìn thấy (VL), và 30 PHz cho cực tím (UV).



Hình 1‑1: Phân loại hệ thống OWC

Nghiên cứu hiện tại về OWC có thể được phân loại thành bốn loại dựa trên sử dụng băng tần : giao tiếp ánh sáng nhìn thấy (VLC) [4], ánh sáng trung thực (Li-Fi) , giao tiếp camera quang học (OCC) [5], và giao tiếp quang học không gian tự do (FSO) [6] như được thể hiện trong Hình 1‑1 . Là một phần của OWC, OCC được coi là một cách thực tế để triển khai VLC trên các thiết bị thông thường và thu hút sự chú ý rộng rãi từ cộng đồng học thuật và công nghiệp. Đặc biệt, sự phát triển gần đây của cảm biến hình ảnh được lắp ráp trong các thiết bị điện tử tiêu dùng phổ cập đã tạo ra cơ hội đáng kể cho việc ứng dụng thực tế của OCC. Trong các hệ thống VLC và LiFi, PD chỉ được sử dụng để nhận dữ liệu. Ngược lại, trong các hệ thống OCC, chức năng giao tiếp của máy ảnh là bổ sung cho chức năng chính của chúng là chụp ảnh. Hệ thống OCC bao gồm các mô-đun truyền và nhận độc lập. Đối với mô-đun truyền, hệ thống OCC tái sử dụng cơ sở hạ tầng đèn điốt phát sáng (LED) được triển khai rộng rãi để tạo ra tín hiệu ánh sáng điều chế. Đối với mô-đun nhận, hệ thống OCC tái sử dụng các thiết bị thông minh (như điện thoại thông minh và máy tính bảng) trang bị camera để bắt gọn tín hiệu ánh sáng tới trong một chuỗi khung hình.

Bảng 1‑1: So sánh hệ thống OCC và VLC

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **VLC** | **OCC** |
| Phạm vi truyền | Thấp | Lên tới vài km |
| Bước sóng | Ánh sáng khả kiến | UV, IR, ánh sáng khả kiến |
| SNR | Thấp | Cao |
| Bộ thu | Photo – dectector | Camera |
| Giải mã | Phức tạp thấp | Phức tạp cao |
| Tốc độ dữ liệu | 11.67kbps – 96Mbps | Thấp hơn VLC |
| Giao thức | IEEE.802.15.7 | IEEE.802.12.7r1 |
| Triển khai MIMO | Khó khăn | Dễ dàng |

Khác với các hệ thống VLC dựa trên photodiod (PD), hệ thống OCC có thể cung cấp sự bao phủ phổ cả trong nhà và ngoài trời bằng cách tận dụng cơ sở hạ tầng hiện có. Ngoài ra, hệ thống OCC cũng có những đặc điểm độc đáo như khả năng tách biệt về không gian và bước sóng, cũng như độ phức tạp thấp và hiệu quả chi phí cao. Chi tiết sự khác biệt giữa hệ thống VLC và OCC được mô tả trong Bảng 1‑1. Do những ưu điểm này, OCC đang nổi lên như một công nghệ không dây quan trọng trong tương lai.

## Tổng quan về hệ thống OCC dựa trên đèn LED

### Nguyên tắc hoạt động hệ thống OCC

Trọng tâm chính của nghiên cứu về công nghệ OCC là tăng tốc độ dữ liệu của nó. Vì tốc độ lấy mẫu của OCC bị giới hạn bởi phần cứng máy ảnh, một số nghiên cứu đã điều tra các cơ hội khác nhau để sử dụng các định dạng điều chế tiên tiến để đạt được tốc độ dữ liệu cao hơn trong các hệ thống OCC. [7] [8]

A diagram of a computer system

Description automatically generated

Hình 1‑2: Sơ đồ khối hệ thống OCC

Như được mô tả trong Hình 1‑2, hệ thống OCC bao gồm ba thành phần chính, đó là máy phát, kênh và máy thu. Trong mô-đun máy phát, dữ liệu có thể được điều chế bằng các kỹ thuật điều chế khác nhau, chẳng hạn như khóa bật-tắt (OOK) [9]. khóa dịch chuyển màu (CSK) [10], điều chế cường độ đa cấp (m-IM) [11], v.v. Sau đó, trình điều khiển LED được sử dụng để kích hoạt đèn LED phát ra ánh sáng điều chế. Trong mô-đun máy thu, các chùm quang khác nhau mang luồng dữ liệu đến máy thu thông qua kênh không dây và được cảm biến hình ảnh chụp và phân biệt đồng thời. Cảm biến hình ảnh ban đầu chuyển đổi tín hiệu ánh sáng tới thành tín hiệu điện, sau đó được lượng tử hóa thành hình ảnh có chứa các sọc và cuối cùng được nén thành một định dạng hình ảnh cụ thể. Sau đó, các sọc giao tiếp có trong hình ảnh được chuyển đổi thành các giá trị thang độ xám bằng phân vùng RoI và lấy mẫu pixel [12]. Bộ giải điều chế cuối cùng được yêu cầu để chuyển đổi các giá trị thang độ xám thành bit.

### Bộ phát

Là một loại của OWC, OCC thường sử dụng tín hiệu ánh sáng nhấp nháy tần số cao từ các thiết bị chiếu sáng hàng ngày để truyền dữ liệu mà mắt thường không thể cảm nhận được. Hiện nay, các thiết bị chiếu sáng được sử dụng phổ biến nhất có đèn LED, đèn huỳnh quang và bóng đèn sợi đốt. Trên các thiết bị chiếu sáng khác nhau, cơ sở hạ tầng chiếu sáng dựa trên LED được triển khai rộng rãi do hiệu quả năng lượng cao, kích thước nhỏ và tuổi thọ dài. Ngoài ra, tốc độ cao của các đặc tính chuyển mạch BẬT / TẮT cũng làm cho đèn LED trở thành lựa chọn ưu tiên cho các máy phát trong các hệ thống OCC. Theo hướng này, các hệ thống OCC có thể truyền tín hiệu trực tiếp bằng cách sử dụng các nguồn sáng LED thương mại, giảm đáng kể chi phí thực hiện và lắp đặt.

Có thể thấy ánh sáng trắng phù hợp hơn với nhu cầu chiếu sáng hàng ngày, vì thế hệ thống OCC thường sử dụng đèn LED trắng làm máy phát. Trên thực tế, ánh sáng trắng là một loại ánh sáng hỗn hợp được hình thành bởi sự pha trộn và chồng chất của một số màu ánh sáng.

Hiện tại có ba loại đèn LED trắng dựa trên nguyên tắc phát sáng lai của chúng: (a) LED xanh + phốt pho vàng (PC-LED). Chip LED màu xanh được sử dụng để kích thích phốt pho màu vàng tạo ra ánh sáng vàng, sau đó ánh sáng xanh và ánh sáng vàng được trộn lẫn để tạo thành ánh sáng trắng. (b) Đèn LED cực tím + phốt pho RGB (RGB + UV-LED). Đèn LED cực tím được phủ một lớp phốt pho ba màu và ánh sáng trắng được tạo ra dưới chiếu xạ cực tím. (c) Đèn LED RGB. Đóng gói các chip LED đơn sắc gồm ba màu cơ bản của ánh sáng (đỏ, xanh lá cây, xanh dương) với nhau để chúng giao nhau để tổng hợp ánh sáng trắng [13].

A diagram of different types of light emitting diffraction

Description automatically generated

Hình 1‑3: Các loại đèn LED trắng

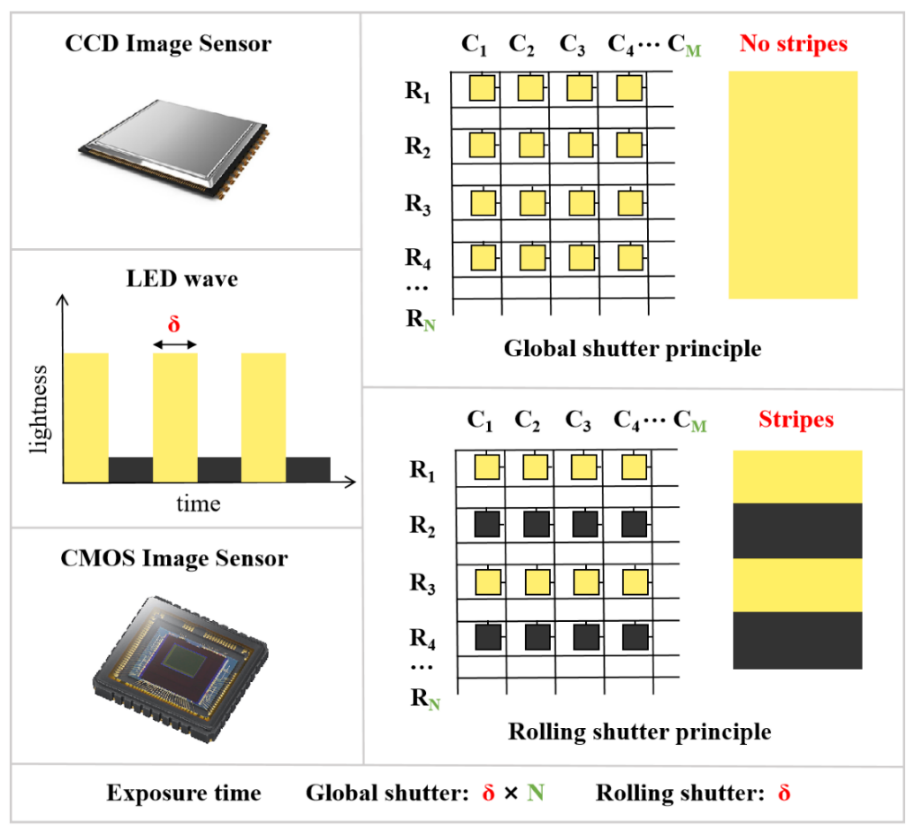
Các loại đèn LED trắng khác nhau có các đặc tính khác nhau có thể ảnh hưởng đến hiệu quả năng lượng của hệ thống. Các loại đèn LED trắng khác nhau có các đặc tính khác nhau có thể ảnh hưởng đến hiệu quả năng lượng của hệ thống, ví dụ, PC-LED, với chi phí thấp và thiết kế đơn giản, nhưng hơi thiếu hiệu quả phát sáng. Đèn LED RGB có thể tránh mất năng lượng của phốt pho trong quá trình chuyển đổi ánh sáng, do đó đạt được hiệu suất phát sáng cao hơn. Tuy nhiên, việc sử dụng nhiều chip LED trong một gói RGB-LED có thể dẫn đến tăng độ phức tạp và tiêu thụ điện năng cao hơn so với PC-LED

### Đầu thu

Khác với các hệ thống VLC dựa trên PD truyền thống, hệ thống OCC sử dụng một camera thương mại dựa trên cảm biến hình ảnh như một bộ cảm biến để nhận dữ liệu, trong đó cấu trúc tiêu biểu của một camera bao gồm ống kính hình ảnh, cảm biến hình ảnh và mạch đọc [14]. Trong đó, ống kính hình ảnh là một thiết bị quang học phức tạp và được sử dụng để tập trung hình ảnh của một đối tượng lên cảm biến hình ảnh. Cảm biến hình ảnh là một mảng hai chiều của các bộ phát hiện ánh sáng được sử dụng để nhận thông tin không gian và sau đó tạo hình ảnh trên màn hình. Do đó, cảm biến hình ảnh có khả năng phân loại nhiều nguồn sáng tách biệt không gian với độ phân giải cao. Ngoài ra, nó có một bộ lọc màu độc lập chiếu sáng nhiều màu sắc, vì vậy nó có thể hoạt động như một bộ thu đa màu tự nhiên. [15]

CCD (Charge-Coupled Devices) và CMOS (Complementary Metal Oxide Semiconductors) là hai loại công nghệ được sử dụng trong máy ảnh kĩ thuật số và các thiết bị hình ảnh khác

Nói chung, CCD yêu cầu nhiều năng lượng hơn cảm biến CMOS, vì chúng sử dụng nguồn năng lượng bên ngoài để chuyển đổi điện tích giữa các điểm ảnh. Mặt khác, cảm biến CMOS sử dụng transistor bên trong để khuếch đại điện tích, giảm thiểu tiêu thụ điện năng. Ngoài ra, cảm biến CMOS thường rẻ hơn so với CCD, vì chúng yêu cầu các quy trình sản xuất ít phức tạp hơn. Điều này làm cho chúng có giá cả phải chăng hơn cho máy ảnh tiêu dùng và các thiết bị chụp ảnh khác.



Hình 1‑4: Sự khác biệt giữa cảm biến CCD và cảm biến CMOS

Cảm biến hình ảnh có hai chế độ hoạt động: hoạt động trập toàn cảnh global shutter và hoạt động trập cuộn (rolling shutter) như được thể hiện trong Hình 1‑4. Trập toàn cảnh thường được sử dụng trong các cảm biến hình ảnh CCD [16], trong khi trập cuộn phù hợp hơn cho các cảm biến hình ảnh CMOS giá rẻ. Ở chế độ trập toàn cảnh, mỗi đơn vị cảm biến ánh sáng đồng bộ thu ánh sáng và được phơi sáng đồng thời. Sau khi phơi sáng hoàn tất, thông tin cho mỗi điểm ảnh được đọc ra tuần tự, cơ chế này có lợi cho việc chụp các vật thể chuyển động, do đó trập toàn cảnh là cách tuyệt vời để có được hình ảnh chất lượng cao cần thiết cho các dịch vụ đa phương tiện. Tuy nhiên, chế độ trập toàn cảnh chỉ ghi lại thông tin được truyền tại một thời điểm nhất định bởi bộ phát. Ở chế độ trập cuộn, thay vì thu ánh sáng đồng thời, quang điện trở quét tuần tự (theo cột hoặc hàng) các điểm ảnh bằng cảm biến hình ảnh CMOS để tạo ra hình ảnh. Trập cuộn mang lại lợi ích cho OCC, vì nhiều trạng thái BẬT/TẮT của nguồn sáng cũng được chụp trong hình ảnh để biểu diễn các bit nhị phân "0" hoặc "1" như được thể hiện trong Hình 1‑5.

Diagram of a diagram of a camera

Description automatically generated with medium confidence

Hình 1‑5: Cảm biến CMOS quét đèn LED ở trạng thái ON và OFF

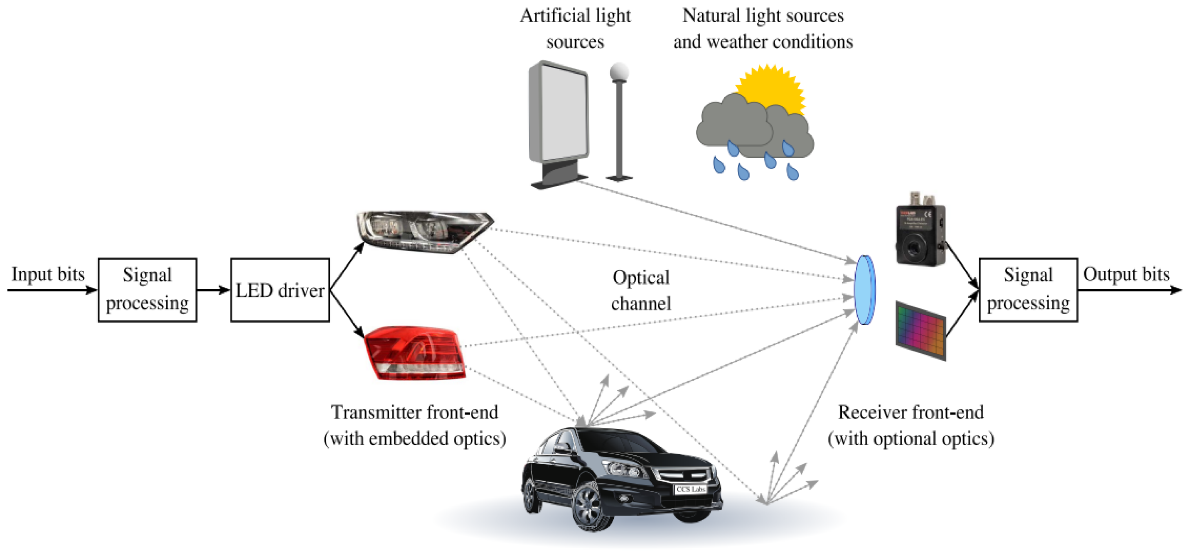
## Các ứng dụng tiềm năng

Hiện nay, các công nghệ truyền thông không dây, như Wifi, Bluetooth và Zigbee, đang trở nên ngày càng phổ biến. Tuy nhiên, mặc dù đã mang lại nhiều tiện ích, nhưng vẫn tồn tại những hạn chế đối với ảnh hưởng đến sức khỏe của trẻ em và người già. Ngoài ra, phân bổ tần số trong dải sóng vô tuyến đang phụ thuộc vào quy định của từng quốc gia và được quản lý bởi các tổ chức viễn thông quốc tế. Cùng với đó, có sự cạnh tranh giữa các nhu cầu dân dụng, quân sự và chi phí đắt đỏ cho hệ thống truyền thông bằng sóng vô tuyến, so với hệ thống OCC (Optical Camera Communication), nơi quang phổ ánh sáng miễn phí hoàn toàn.

Tính chất lan truyền của ánh sáng mang lại nhiều ưu điểm về bảo mật so với sóng vô tuyến. Sóng vô tuyến có khả năng lan truyền xa, đi qua các vật cản và bức tường, tuy nhiên, điều này có thể gây nguy hiểm cho tính bảo mật. Ngược lại, ánh sáng chỉ lan truyền trong không gian mà nó có thể nhìn thấy, tạo ra một môi trường bảo mật an toàn hơn.

### Hệ thống giao thông thông minh dựa trên OCC

Môi trường ngoài trời hiện tại có một lượng đáng kể cơ sở hạ tầng LED, và hầu hết các phương tiện đều được trang bị camera tích hợp, làm cho OCC trở thành một công nghệ truyền thông hết sức hấp dẫn. Đèn LED có đặc tính tiêu thụ điện năng thấp và thời gian phản ứng ngắn. Khả năng phân tách không gian của cảm biến hình ảnh tích hợp trong camera mang lại khả năng chống nhiễu hiệu quả đối với các tác động giao thoa. Những đặc tính đã đề cập là quan trọng đối với truyền thông xe cộ, khiến cho OCC trở nên đáng kể hơn nhiều so với các công nghệ truyền thông không dây khác.



Hình 1‑6: Hệ thống giao thông thông minh dựa trên OCC

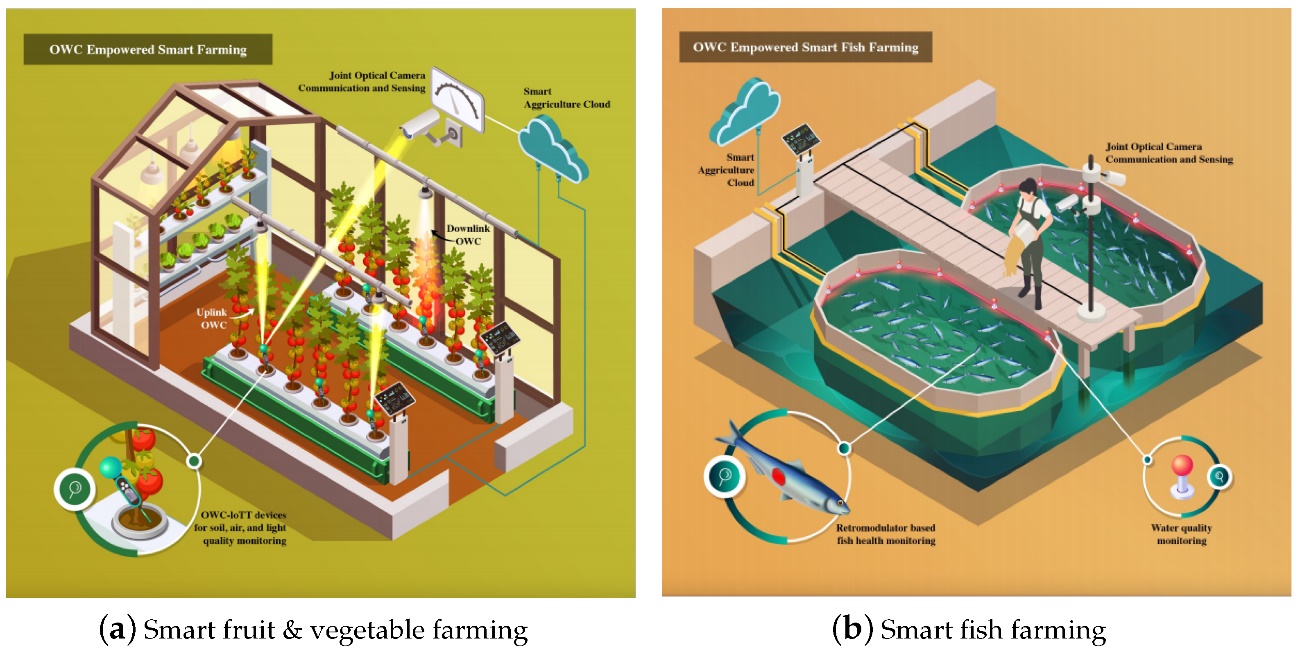
Cơ sở hạ tầng nằm dọc theo các con đường, chẳng hạn như tín hiệu giao thông, đèn pha xe gần đó, đèn phanh và đèn đường, sẽ gửi dữ liệu được nhận bởi các cảm biến hình ảnh được cài đặt trên xe [20].Do đó, các hệ thống giao thông thông minh dựa trên OCC (ITSs) cho phép giao tiếp giữa xe với cơ sở hạ tầng (V2I), cơ sở hạ tầng với xe (I2V) và xe với xe (V2V).

Trong ITSs (Intelligent transportation systems) dựa trên OCC, dữ liệu được truyền qua đèn LED bao gồm các thông số như kinh độ, vĩ độ và thông tin liên quan đến an toàn của xe. Hơn nữa, dữ liệu hỗ trợ lái xe có thể được truyền từ xe này sang xe khác để tăng cường an toàn và trải nghiệm lái xe. Trong những năm gần đây, các hệ thống OCC của xe đã cho thấy hứa hẹn to lớn trong việc tăng cường hiệu quả và an ninh giao thông. Ví dụ, vào năm 2014, P. Ji et al. đã nghiên cứu khả năng sử dụng đèn LED hiện có trong xe hoặc đèn giao thông và camera của điện thoại thông minh để thực hiện VLC xe cộ (V2LC) [21]. Vào năm 2023, K. Xu và cộng sự đã tiết lộ dự án mới nhất của mình, NeuromorphicVLC. Nguyên mẫu này được trang bị camera neuromorphic nhằm mục đích giao tiếp và kết nối mạng xe cộ [22]. Những nghiên cứu trên đã góp phần thúc đẩy việc thực hiện nó trong các ứng dụng thực tế.

### Kết nối IoT dựa trên OCC

Internet of Things, thường được gọi là IoT, đại diện cho mạng lưới các thiết bị và cảm biến vật lý trong môi trường thông minh. Sự kết nối này cho phép các đối tượng giao tiếp và trao đổi dữ liệu giữa chúng. Ngày nay, IoT đã chứng minh tiềm năng to lớn của nó bằng cách cung cấp khả năng cho hàng tỷ kết nối thiết bị trên toàn thế giới. Mô hình IoT mở ra cánh cửa cho sự đổi mới, cho phép hiện thực hóa các thành phố, cơ sở hạ tầng và dịch vụ thông minh để nâng cao chất lượng cuộc sống và cải thiện việc sử dụng tài nguyên. [23]

OCC với phổ tần số khổng lồ được tích hợp với IoT có thể cung cấp khả năng mở rộng cho các ứng dụng trong nhà và ngoài trời trong các môi trường thông minh trong tương lai như thành phố thông minh, nhà thông minh, nông nghiệp thông minh và nhà máy thông minh, như thể hiện trong Hình 1-9.



Hình 1‑7: Nông nghiệp thông minh dựa trên OCC – IoT

Trong mạng IoT, việc sử dụng camera trong các hệ thống OCC cho phép thu thập và giám sát hiệu quả dữ liệu phát ra từ các thiết bị và cảm biến khác nhau. Sau khi nhận dữ liệu từ nhiều thiết bị, hệ thống OCC sẽ xử lý thông tin và truyền đến cổng bằng mạng có dây hoặc không dây. Dữ liệu sẽ được chuyển tiếp đến các mạng, chẳng hạn như 5G hoặc mạng vệ tinh, sau đó sẽ được lưu trữ trong máy chủ hoặc đám mây để kích hoạt các ứng dụng dựa trên IoT.

### Định vị trong nhà (IPS)

OCC mang đến tiềm năng lớn không chỉ trong việc truyền dữ liệu không dây thông thường mà còn đặc biệt là trong các môi trường trong nhà. Ở những nơi như vậy, ánh sáng có thể được kiểm soát một cách hiệu quả và ít bị ảnh hưởng bởi các yếu tố bên ngoài. Sự phổ biến ngày càng cao của đèn LED và sự gia tăng đột biến của số lượng điện thoại di động trên toàn thế giới dự kiến sẽ tiếp tục đẩy mạnh nhu cầu sử dụng đèn LED trong thời gian sắp tới.

A graph of led lighting market

Description automatically generated

Hình 1‑8: Nhu cầu sử dụng đèn LED đến năm 2030

Vì vậy, OCC 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.

Với sự phổ biến của công nghệ chiếu sáng LED và sự phát triển nhanh chóng của cảm biến CMOS độ phân giải cao, các hệ thống định vị trong nhà dựa trên OCC (IPS) đã mở ra sự phát triển mạnh mẽ và triển vọng rộng lớn [17].

Bảng 1‑2: So sánh các công nghệ IPS hiện có

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Công nghệ IPS** | **Sai số (m)** | **Ưu điểm** | **Hạn chế** | **Tận dụng cơ sở hạ tầng** |
| GPS | 6-20 | Phủ sóng khắp thế giới | Thích hợp ngoài trời, yêu cầu chi phí cơ sở hạ tầng cao | Có |
| Wi-Fi | 1-5 | Tận dụng cơ sở hạ tầng hiện có | Phụ thuộc cao vào môi trường, chi phí triển khai cao | Có |
| Bluetooth | 1-5 | Chi phí và tiêu thụ năng lượng thấp | Độ chính xác thấp, phạm vi nhỏ, dễ nhiễu | Không |
| UWB và Ultrasound | 0.1-1 | Định vị chính xác, hạn chế nhiễu | Phạm vi phủ sóng thấp, chi phí cao | Không |
| RFID | 1-2 | Tiêu thụ năng lượng thấp, phạm vi rộng | Thời gian phản hồi cao, độ chính xác thấp, khả năng di động bị hạn chế | Không |
| Infrared | 1-2 | Tiêu thụ năng lượng thấp | Phạm vi thấp, yêu cầu đèn LED hồng ngoại đặc biệt | Không |
| OCC | 0.01-0.32 | Chi phí thấp, độ chính xác cao, cho phép tận dụng hệ thống LED sẵn có | Cần can thiệp vào cơ sở hạ tầng hiện có để sử dụng, tầm hoạt động có thể bị hạn chế bởi vật cản | Có |

So với các hệ thống định vị trong nhà khác sử dụng RF, chẳng hạn như nhận dạng RF (RFID), mạng cục bộ không dây (WiFi) và công nghệ định vị Bluetooth, hệ thống định vị trong nhà dựa trên ánh sáng khả kiến có một số lợi thế [18]. Chúng bao gồm mức độ chính xác cao trong định vị, khả năng miễn nhiễm với nhiễu điện từ, chi phí thấp và khả năng thực hiện các nhiệm vụ định vị và chiếu sáng đồng thời. Đặc biệt, độ chính xác định vị của IPS dựa trên OCC nằm trong phạm vi cm [19], vượt trội so với công nghệ định vị dựa trên RF.

Hệ thống định vị trong nhà sử dụng truyền thông không dây quang học LED và camera (OCC) là một phương pháp thông minh để xác định vị trí trong không gian bằng cách tận dụng ánh sáng của các đèn LED. Dưới đây là giải thích cách hệ thống này hoạt động:

* **Phát tín hiệu từ đèn LED:**

Các đèn LED được đặt tại các vị trí cố định trong không gian cần định vị, chẳng hạn như trần nhà hoặc tường. Mỗi đèn LED phát tín hiệu quang học có thể mang một định danh duy nhất (ID) hoặc một chuỗi dữ liệu đặc biệt.

* **Thu thập dữ liệu từ Camera:**

Khi người dùng hướng điện thoại về một đèn LED, camera trên điện thoại sẽ thu thập ánh sáng từ đèn LED đó.Ánh sáng này có thể chứa thông tin về ID hoặc dữ liệu được truyền từ đèn LED.

* **Truy vấn cơ sở dữ liệu:**

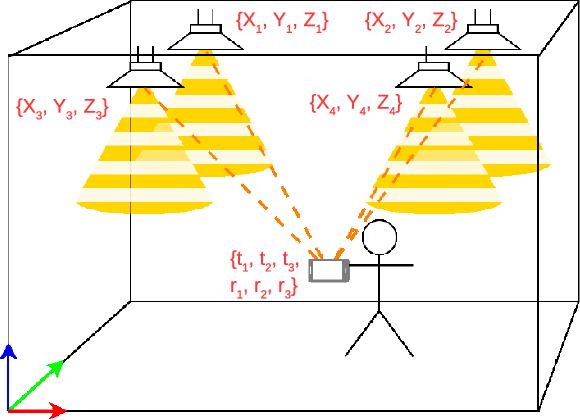
Hệ thống sẽ sử dụng thông tin thu thập được từ camera để truy vấn vào cơ sở dữ liệu. Cơ sở dữ liệu chứa thông tin về vị trí của các đèn LED và ID tương ứng.

* **Xác định vị trí:**

Dựa trên ID hoặc dữ liệu từ đèn LED được truyền, hệ thống xác định vị trí hiện tại của điện thoại trong không gian.

* **Hướng dẫn điều hướng:**

Sau khi xác định vị trí, hệ thống có thể cung cấp hướng dẫn điều hướng đến vị trí mong muốn thông qua ứng dụng điện thoại. Hướng dẫn có thể hiển thị trực tiếp trên màn hình điện thoại hoặc thông qua âm thanh và rung.



Hình 1‑9: Hệ thống định vị trong nhà dựa trên OCC

# TỔNG QUAN KĨ THUẬT SỬ DỤNG TRONG HỆ THỐNG OCC

## Các kĩ thuật điều chế cho hệ thống LED- Camera

Có nhiều kỹ thuật điều chế được phát triển cho OCC, như variable pulse position modulation (VPPM), spatial 2-phase shift keying (S2-PSK), và pulse width modulation (PWM),v.v... Tuy nhiên, rào cản chính cho tất cả các kỹ thuật điều chế dựa trên OCC là hiện tượng nhấp nháy. Các kỹ thuật điều chế hiện tại không bị nhấp nháy bị hạn chế bởi việc cung cấp một máy ảnh ở khoảng cách gần.

Bảng 2‑1 tóm tắt các phương pháp điều chế được đề xuất và thực hiện cho OCC cùng với các đặc điểm của chúng. Việc triển khai các phương pháp biến đổi khác nhau sẽ đòi hỏi các yêu cầu khác nhau. [24]

Bảng 2‑1: Các phương pháp điều chế đề xuất

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kĩ thuật điều chế | RGB LED | Lấy mẫu dưới | Tốc độ khung hình (fps) | Tốc độ dữ liệu | Khoảng cách | Ứng dụng |
| OOK |  |  | 28 | 896 bps | 25 cm | LBS |
| UFSOOK |  |  | 30 | 15 bps  240 bps | 40 cm  50 cm | V2X, LBS |
| CSK |  |  | 30 | 5.2 kbps | 3 m | LBS |
| UFSOOK |  |  | 30 | 150 bps | 60 m | V2X |
| UQAMSM |  |  | 50 | 500 bps | 1.5 m | LBS |
| m-IM |  |  | 50 | >10 kbps | 2 m | LBS, Bảng điện tử |
| UPWM |  |  | 30 | 150 bps | 1 m | LBS |
| DCO-OFDM |  |  | 30 | 55 Mbps | 1.5 m | V2X, AR |
| CSK và mutilevel PAM |  |  | 330 | 95 kbps | 1.2 m | V2X, Truyền dữ liệu lớn |

### Kĩ thuật điều chế On Off Keying (OOK)

Phương thức truyền dẫn tín hiệu quang(OCC) thường sử dụng OOK là phương pháp biến đổi tín hiệu phổ biến.

Diagram of a diagram of a light source

Description automatically generated

Hình 2‑1: On-off keying (OOK bằng đèn LED

Hình 2‑1 cho thấy cách điều chế OOK dựa trên “BẬT” và “TẮT” của đèn LED. Như hình minh họa, đèn LED “BẬT” khi dữ liệu “1” được truyền đi, trong khi nó chuyển sang “TẮT” khi dữ liệu “0” được truyền đi. Lợi thế chính của việc sử dụng OOK là thiết kế và triển khai dễ dàng. đại diện cho chu kỳ cơ bản của tín hiệu được truyền .

A diagram of a number and a line

Description automatically generated with medium confidence

Hình 2‑2: Loại bỏ hiện tượng nhấp nháy bằng mã hoá Manchester

Tuy nhiên, việc sử dụng chỉ OOK sẽ gây hiện tượng nhấp nháy khi biểu diễn các chuỗi dài của "0" hoặc "1". Để khắc phục hiện tượng này, mã hóa Manchester (ME) được áp dụng và truyền đi để ngăn chặn hiện tượng nhấp nháy của đèn đường LED được mắt người nhận ra. ME được thiết lập để đảo ngược tín hiệu bằng cách thay đổi dữ liệu “1” thành “1, 0” và dữ liệu “0” thành “0, 1”. Tốc độ dữ liệu được giảm xuống hai lần so với OOK. [25]

A diagram of data processing

Description automatically generated

Hình 2‑3: Cấu trúc khung dữ liệu: (a) Khung dữ liệu (b) Khung truyền có mã hóa manchester

Để đạt được tốc độ truyền dữ liệu cao, đèn LED cần có tần suất nhấp nháy cao. Hơn nữa, máy ảnh cũng cần có khả năng nhận luồng dữ liệu, dẫn đến sự cần thiết cho máy ảnh có tốc độ lấy mẫu cao.

A different types of sound waves

Description automatically generated with medium confidence

Hình 2‑4: Các khung hình được chụp của điều chế OOK với các tần số khác nhau

## Kĩ thuật điều chế MPPM

### Giới thiệu

Từ tài liệu được trích dẫn [26] [27], chủ yếu trọng tâm chính của các nghiên cứu trong lĩnh vực này là cải thiện công suất hoặc hiệu quả băng thông, nhưng lại ít chú ý đến việc kiểm soát độ sáng. Điều này cũng đúng do thực tế là các sơ đồ điều chế để truyền dữ liệu ban đầu được thiết kế cho truyền thông quang học trong không gian tự do dựa trên truyền dẫn hồng ngoại (IR) và không yêu cầu bất kỳ cơ chế kiểm soát độ sáng nào.

Tài liệu [28] giới thiệu kỹ thuật điều chế MPPM (Multi-Pulse Position Modulation) biến tốc độ để đạt được điều khiển đồng thời độ sáng và tốc độ truyền dữ liệu trong hệ thống truyền thông ánh sáng. Nó cung cấp một giải pháp linh hoạt nhưng hiệu quả để đạt được kiểm soát độ sáng và truyền dữ liệu.

Trong MPPM, nhiều xung được truyền, và thông tin được biểu diễn bằng các tổ hợp khác nhau của vị trí các xung này.

Bảng 2‑2: Ví dụ cho điều chế MPPM với n=8

|  |  |  |  |
| --- | --- | --- | --- |
| Tham số | Chỉ số độ sáng | Số lượng kí tự mã hoá ( | Bits/kí tự ] |
| 1 | 0.125 | 8 | 3 |
| 2 | 0.250 | 28 | 4 |
| 3 | 0.375 | 56 | 5 |
| 4 | 0.500 | 70 | 6 |
| 5 | 0.625 | 56 | 5 |
| 6 | 0.750 | 28 | 4 |
| 7 | 0.875 | 8 | 3 |

PPM là trường hợp đặc biệt của MPPM chỉ sử dụng một khe xung trên mỗi khối tín hiệu

Khả năng mang thông tin của MPPM bị ảnh hưởng bởi hai yếu tố:

1. tổng số vị trí trên khối tín hiệu
2. số lượng khe xung quang học trên khối tín hiệu

A diagram of a signal block

Description automatically generated

Hình 2‑5: Khối tín hiệu trong MPPM

Số lượng kí tự (symbol) khác nhau tối đa có thể mã hoá được tính là với Codeword được định nghĩa là một chuỗi các ký tự (bits) được sử dụng để biểu diễn một đơn vị thông tin . Số bit thông tin (codeword) tối đa truyền trên mỗi khối tín hiệu được cho bởi: trong khi PPM thông thường truyền bit. Điều này dẫn đến việc giảm băng thông bởi MPPM với cùng hiệu suất truyền dẫn. Với là số xung trong mỗi khối tín hiệu, băng thông giảm xuống còn so với PPM. [29]

Đối với một giá trị cố định của tham số , số lượng ký tự có thể được mã hóa phụ thuộc vào lựa chọn hiện tại của tham số , được điều chỉnh để đạt được mức độ sáng mong muốn. Vì tỷ lệ mã hóa có thể đạt được không giống nhau ở các mức độ sáng khác nhau. Lựa chọn tham số đặt ra các giới hạn cho mức độ sáng có thể đạt được, trong đó tương ứng với độ sáng tối thiểu, trong khi cung cấp độ sáng tối đa. Các thông số khác nhau liên quan đến điều chế MPPM được cung cấp trong Bảng 1 với .

### Bộ mã hoá

Để minh họa quy trình mã hóa/giải mã, trước hết chúng ta xây dựng Bảng 2‑3, trong đó mỗi hàng tương ứng với một giá trị cụ thể của tham số . Đối với hàng thứ r và cột thứ với và , giá trị tương ứng trong Bảng 3 được thu được như sau:

Bảng 2‑3: Ma trận mã hóa và giải mã ký hiệu

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | 0 |  | 0 |  |  |
|  |  |  |  | 0 |  | 0 |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 0 |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Ví dụ minh hoạ với ta được kết quả của Bảng 2‑4 như sau:

Bảng 2‑4: Ví dụ minh họa mã hóa kí tự

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  | 0 |  |
|  |  |  |  |  |  |
|  | 6 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Đặt chỉ số độ sáng mong muốn và ký tự mã hoá , chúng ta cần sử dụng codeword tương ứng . Quá trình mã hóa tuần theo MSB truyền đầu tiên diễn ra như sau:

1. Bắt đầu từ cột, hàng , giá trị ô là nên chuyển sang cột
2. Cột , hàng, giá trị ô là nên trừ đi giá trị ô: = 2 - 1 = 1
3. Tiếp tục xuống cột , hàng r = 2, giá trị ô là , nên trừ đi giá trị ô:
4. Xuống cột , hàng, giá trị ô là 1 nhưng không trừ nữa mà giữ nguyên vì giá trị không thể âm
5. Hiện tại mới có 2 bit 1 trong codeword, cần thêm 1 bit 1 nữa để đạt Gán thêm 1 cho vị trí cột , hàng
6. Như vậy, codeword tương ứng là

Đối với ví dụ này các chỉ số độ sáng có thể là 0.2, 0.4, 0.6 và 0.8. Một tập hợp các codeword cho các giá trị khác nhau của tham số và , có thể được sử dụng để mã hóa nhị phân, được lập bảng trong Bảng 2‑5.

Bảng 2‑5: Chuỗi mã hóa cho n = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Quá trình mã hoá được thể hiện trong **Thuật toán 1**. Đối với một chỉ số độ sáng nhất định,thuật toán mã hóa được đề xuất thực hiện một ánh xạ duy nhất từ số thập phân sang tập chữ số hệ nhị phân trong đó có bit là 1

|  |
| --- |
| **Thuật toán 1:** Triển khai cho Bộ mã hóa |
| **Input:** Biến  **Output:** Codeword mã hoá  **while** **do**  **| if then**  **| |**  **| else**  **| |\_**  **| if**  **then**  **| |**  **| |**  **| |**  **else**  **| |\_**  **|\_** |
|  |

### Bộ giải mã

Việc triển khai bộ giải mã theo thứ tự từ MSB đến LSB dựa trên giá trị vị trí của các bit 1 yêu cầu đầu vào codeword và tham số . Duyệt codeword từ vị trí cuối cùng đến đầu . Theo ví dụ minh họa cho giải điều chế chogiá trị kí tự:

Quá trình giải mã được thể hiện trong **Thuật toán 2**

|  |
| --- |
| **Thuật toán 2:** Thuật toán giải điều chế |
| **Input:** Codeword có độ dài  **Output:** Kí tự dữ liệu  **For** **to**  **do**  **| if then**  **| | if**  **then**  **| | |\_**  **| |**\_  **|**\_ |

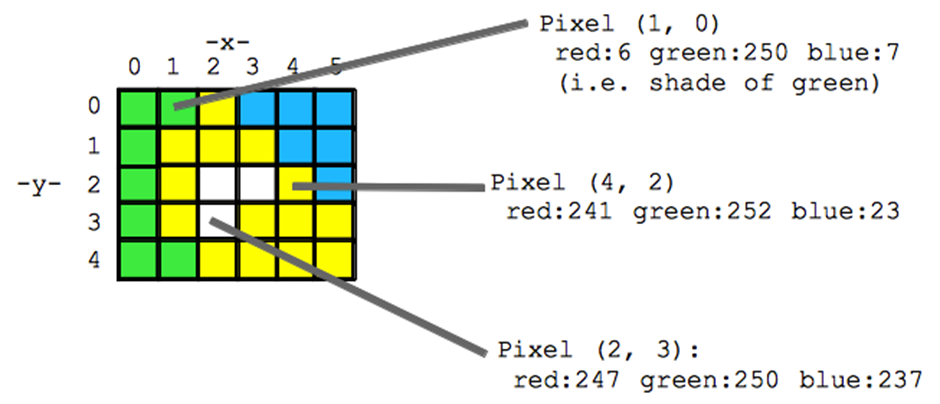
## Thị giác máy trên thiết bị di động.

### Tổng quan về ảnh kỹ thuật số

#### **Điểm ảnh (Picture Element)**

Điểm ảnh thường được gọi là pixel, là đơn vị nhỏ nhất của hình ảnh máy tính. Chúng là những chấm màu riêng lẻ tạo thành một bức tranh khi kết hợp. Các pixel được tổ chức theo thiết kế dạng lưới, trong đó mỗi pixel nói lên một điểm dữ liệu màu duy nhất. Những trọng tâm này cùng nhau làm cho đại diện trực quan của bức tranh.

Mỗi pixel trong một hình ảnh kỹ thuật số chứa dữ liệu cụ thể về màu sắc và nồng độ của nó. Màu sắc của một pixel thường được biểu diễn bằng cách sử dụng màu RGB (Red, Green, Blue), trong đó mỗi kênh màu được phân bổ một giá trị từ 0 đến 255. Bằng cách kết hợp các cường độ khác nhau của ba màu thiết yếu này, có thể đạt được nhiều màu sắc



Hình 2‑6: Toạ độ điểm ảnh và màu sắc

Trong Hình 2‑6, ví dụ về 3 pixel lần lượt có vị trí tọa độ (1,0), (2,3) và (4,2). Mỗi pixel sẽ có giá trị tương ứng để biểu thị độ sáng của nó. Điều này phụ thuộc vào loại hình ảnh được sử dụng. Trong lĩnh vực ảnh kỹ thuật số, có 3 loại chính mà chúng ta có thể xử lý, đó là ảnh nhị phân, ảnh xám và ảnh màu. Mỗi loại hình ảnh mang đến những đặc điểm độc đáo và yêu cầu cách xử lý khác nhau.

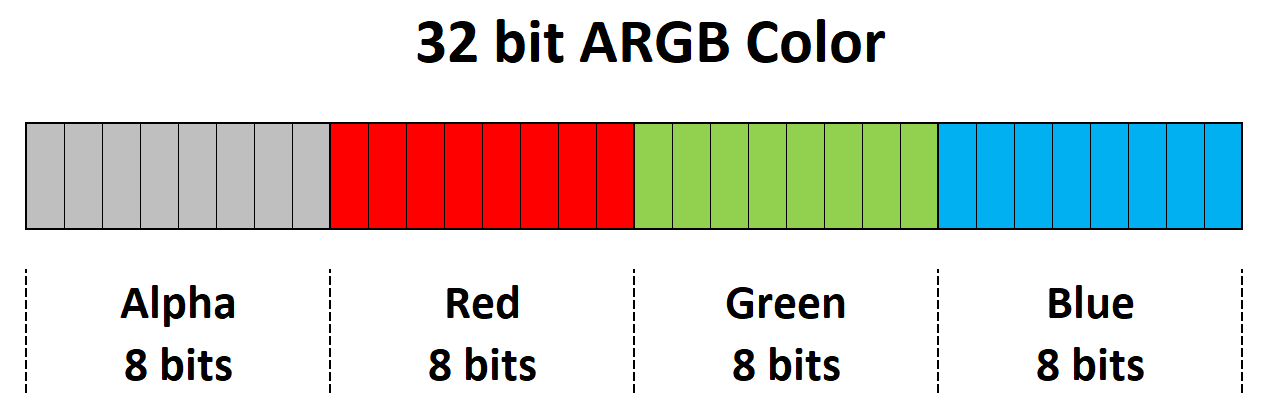
#### **Ảnh màu**

Pixel cho phép thể hiện màu sắc chính xác trong hình ảnh kỹ thuật số. Bằng cách thay đổi cường độ của mỗi kênh màu, có thể đạt được rất nhiều màu sắc. Khả năng thích ứng này cho phép tạo màu thực tế và linh hoạt trong các ứng dụng khác nhau.

Bảng 2‑6: Biểu đồ màu

|  |  |  |  |
| --- | --- | --- | --- |
| Màu sắc | Cường độ kênh đỏ | Cường độ kênh lục | Cường độ kênh lam |
| Đỏ | 255 | 0 | 0 |
| Tím | 255 | 0 | 255 |
| Vàng | 255 | 255 | 0 |
| Cam | 100 | 100 | 0 |
| Trắng | 255 | 255 | 255 |
| Đen | 0 | 0 | 0 |

Ảnh màu là ảnh kỹ thuật số được biểu thị thông qua ba giá trị màu cơ bản: R (Red), G (Green) và B (Blue). Ba giá trị màu này biểu thị màu sắc và độ sáng của mỗi điểm ảnh. Trong lĩnh vực hình ảnh kỹ thuật số, chúng được kết hợp để tạo ra giá trị của một điểm ảnh cụ thể, như được minh họa trong Bảng 2‑6. Cách giá trị màu được biểu diễn và lưu trữ có thể làm cho ảnh màu được lưu trữ dưới dạng 24 bits hoặc 32 bits. Không gian màu RGB 32-bit thường được sử dụng để biểu diễn mỗi điểm ảnh với mức sâu màu sắc 32 bit. Điều này thường bao gồm 8 bit cho mỗi thành phần màu (đỏ, xanh lá cây, xanh lam) và 8 bit thêm cho kênh alpha được biểu thị như Hình 2‑7.



Hình 2‑7: Ảnh màu 32 bits

#### **Ảnh xám (gray image)**

Một ảnh xám (hoặc cấp độ xám) đơn giản là một ảnh trong đó chỉ có các màu là các tông xám. Lý do để phân biệt các hình ảnh như vậy với bất kỳ loại hình ảnh màu nào khác là vì cần cung cấp ít thông tin hơn cho mỗi điểm ảnh. Trong thực tế, một màu 'xám' là một màu mà các thành phần đỏ, xanh lá cây và xanh dương có cường độ bằng nhau trong không gian màu RGB, và do đó chỉ cần xác định một giá trị cường độ duy nhất cho mỗi điểm ảnh, không giống như ba cường độ cần thiết để chỉ định mỗi điểm ảnh trong một hình ảnh màu đầy đủ.

Cường độ xám được lưu trữ dưới dạng số nguyên 8 bit, cung cấp 256 tông xám khác nhau từ đen đến trắng. Hình ảnh thang độ xám thường được sử dụng trong các ứng dụng khác nhau, chẳng hạn như nhiếp ảnh, hình ảnh y tế, thị giác máy tính và xử lý hình ảnh, khi thông tin màu sắc không cần thiết hoặc khi đơn giản hóa việc thể hiện hình ảnh. Những hình ảnh này đơn giản hơn để xử lý và yêu cầu lưu trữ dữ liệu ít hơn so với hình ảnh đầy đủ màu sắc, làm cho chúng hữu ích cho một số tác vụ nhất định.. Ở Hình 2‑8 tương ứng từng vị trí của điểm ảnh là giá trị của nó.

A grid with a red arrow pointing to a city

Description automatically generated

Hình 2‑8: Ảnh xám 8 bit

#### **Ảnh nhị phân (binary image)**

Ảnh nhị phân (binary image) là một ảnh mà mỗi pixel chỉ có một trong hai giá trị là 0 hoặc 1.

A diagram of a number of objects

Description automatically generated with medium confidence

Hình 2‑9: Ảnh nhị phân áp dụng ngưỡng

Ta thường sử dụng ảnh xám để tạo ảnh nhị phân. Quá trình biến đổi từ một ảnh xám sang ảnh nhị phân (Hình 2‑9) thì sẽ cần đến một giá trị ngưỡng (threshold). Ngưỡng (threshold) là một giá trị xác định ranh giới giữa các pixel đen và trắng. Các pixel có giá trị lớn hơn ngưỡng sẽ được gán giá trị 1 (trắng), còn nhỏ hơn ngưỡng sẽ được gán giá trị 0 (đen).

A collage of images of a statue

Description automatically generated

Hình 2‑10: Ảnh nhị phân với từng giá trị ngưỡng khác nhau

Với mỗi giá trị threshold thì sẽ cho ra một ảnh nhị phân khác nhau, được mô tả như Hình 2‑10. Việc áp dụng ngưỡng giúp phân tách các đối tượng ra khỏi nền trong ảnh. Nó thường được dùng trong xử lý ảnh để làm nổi bật các đặc trưng của đối tượng.

Có nhiều phương pháp xác định ngưỡng như ngưỡng cố định (fixed threshold), ngưỡng tự động Otsu, ngưỡng làm mịn (smoothing threshold)... tùy thuộc vào ứng dụng. Ảnh nhị phân thường được sử dụng trong các bài toán phân đoạn ảnh, nhận dạng đối tượng, tách chữ và nền...

### Thị giác máy trong VLC

#### **Định nghĩa**

Thị giác máy (MV) là một lĩnh vực của trí tuệ nhân tạo (AI) cho phép máy tính và hệ thống lấy thông tin hữu ích từ hình ảnh kỹ thuật số, video và các đầu vào trực quan khác. Trên điện thoại thông minh, MV được sử dụng trong một loạt các ứng dụng, bao gồm:

* **Nhận diện đối tượng**: MV được sử dụng để phát hiện và nhận dạng các đối tượng trong môi trường xung quanh của điện thoại. Điều này có thể được sử dụng cho các ứng dụng như nhận dạng khuôn mặt, phát hiện đối tượng và điều hướng AR.



Hình 2‑11: Nhận diện khuôn mặt trên iPhone X bằng công nghệ Face ID

* **Phân tích hình ảnh**: MV được sử dụng để phân tích hình ảnh để tìm các mẫu hoặc thông tin cụ thể. Điều này có thể được sử dụng cho các ứng dụng như chỉnh sửa ảnh, dịch ngôn ngữ và phân tích thị trường.

Thị giác máy đang trở nên ngày càng phổ biến trên điện thoại thông minh do sự cải thiện của phần cứng và phần mềm. Các cảm biến camera hiện đại có độ phân giải cao hơn và khả năng thu sáng tốt hơn, trong khi các thuật toán MV ngày càng chính xác hơn. Điều này cho phép các nhà phát triển ứng dụng tạo ra các ứng dụng MV mạnh mẽ và sáng tạo hơn.

#### **Ưu điểm**

Dưới đây là một số ưu điểm của thị giác máy trên các điện thoại di động thông minh:

* **Tính tiện dụng:** Thị giác máy có thể được sử dụng để thực hiện các tác vụ trên điện thoại thông minh mà không cần nhập thông tin thủ công. Điều này có thể giúp tiết kiệm thời gian và công sức cho người dùng. Ví dụ: tính năng nhận dạng khuôn mặt trên điện thoại thông minh cho phép người dùng mở khóa điện thoại của họ mà không cần nhập mật khẩu.
* **Tính tương tác:** Thị giác máy có thể được sử dụng để tạo ra các trải nghiệm tương tác hơn trên điện thoại thông minh. Ví dụ: các ứng dụng AR sử dụng thị giác máy để tạo ra các thế giới ảo mà người dùng có thể tương tác.
* **Tính chính xác:** Thị giác máy ngày càng trở nên chính xác hơn. Điều này có nghĩa là các ứng dụng MV có thể cung cấp thông tin và trải nghiệm chính xác hơn. Ví dụ: các ứng dụng dịch ngôn ngữ sử dụng thị giác máy để dịch văn bản trong thời gian thực.

#### **Ngôn ngữ, thư viện và nền tảng hỗ trợ**

Để hỗ trợ công nghệ thị giác máy trên điện thoại di động thông minh, có một số ngôn ngữ lập trình và thư viện xử lý ảnh phổ biến mà các nhà phát triển sử dụng. Dưới đây là một số ngôn ngữ và thư viện quan trọng:

**Ngôn ngữ Lập trình:**

* Java: Được sử dụng rộng rãi cho phát triển ứng dụng Android. Android Studio là môi trường phát triển tích hợp chính cho Java trên Android.
* Swift và Objective-C: Được sử dụng cho việc phát triển ứng dụng trên iOS. Xcode là môi trường phát triển chính cho cả Swift và Objective-C.
* Kotlin: Là một ngôn ngữ lập trình mới được Google đề xuất cho phát triển ứng dụng Android và có thể thay thế cho Java.
* C#: Được sử dụng trong môi trường phát triển ứng dụng Windows Phone và cũng có thể được sử dụng cho phát triển ứng dụng trên nền tảng Xamarin cho cả Android và iOS.

**Thư viện và Framework Xử lý ảnh:**

* OpenCV: Thư viện mã nguồn mở phổ biến nhất để xử lý ảnh và thị giác máy tính. Hỗ trợ nhiều thuật toán xử lý ảnh cơ bản và nâng cao.
* Matplotlib: Thư viện vẽ đồ thị phổ biến trong Python. Có thể dùng để hiển thị và xử lý ảnh.
* PIL/Pillow: Thư viện xử lý ảnh Python. Cung cấp các phương thức để thao tác ảnh như cắt, xoay, thay đổi kích thước, lưu định dạng...
* Scikit-Image: Thư viện xử lý ảnh trong Python dựa trên NumPy và SciPy.
* TensorFlow: Thư viện học máy nổi tiếng. Có mô hình và các API xử lý ảnh tiên tiến.
* Keras: API để xây dựng mô hình học sâu trên TensorFlow hoặc CNTK. Dễ sử dụng.
* PyTorch: Thư viện học sâu phổ biến với nhiều mô hình xử lý ảnh hiệu quả cao.
* MATLAB: Phần mềm mạnh mẽ xử lý ảnh và tính toán kỹ thuật.
* SimpleCV: Framework mã nguồn mở xử lý ảnh trên Python.Các ngôn ngữ và thư viện này đều hỗ trợ việc phát triển ứng dụng thị giác máy và xử lý ảnh trên điện thoại di động thông minh, giúp tạo ra các trải nghiệm người dùng độc đáo và sáng tạo.

Trong đồ án này, em sẽ sử dụng hai ngôn ngữ lập trình là **Python** và **Kotlin** để xây dựng ứng dụng di động về xử lý ảnh.

Lý do lựa chọn Python là bởi ngôn ngữ này có nhiều thư viện mạnh mẽ để xử lý ảnh và thị giác máy tính như OpenCV, NumPy, SciPy... Những thư viện này hỗ trợ rất nhiều thuật toán xử lý ảnh tiên tiến, giúp Python trở thành một công cụ lý tưởng để thực hiện các tác vụ phân tích và xử lý ảnh.

Trong khi đó, Kotlin là ngôn ngữ được hỗ trợ chính thức trong **Android SDK**. Nó cũng là ngôn ngữ hiện đại, linh hoạt để phát triển các ứng dụng di động đa nền tảng. Do đó em chọn Kotlin để xây dựng giao diện người dùng và tích hợp chức năng xử lý ảnh của Python vào ứng dụng Android.

Để kết hợp hai ngôn ngữ này, em sẽ sử dụng Android Studio làm IDE và tích hợp Python thông qua thư viện **Chaquopy**. Chaquopy là một plugin cho Android Studio, giúp tích hợp Python vào ứng dụng Android. Nó cung cấp một cách thuận tiện để phát triển ứng dụng sử dụng cả Java (hoặc Kotlin) và Python. Chaquopy giúp kết hợp sức mạnh của Python trong việc xử lý dữ liệu và tính toán với khả năng phát triển ứng dụng Android một cách hiệu quả. Điều này đặc biệt hữu ích đối với các nhà phân tích dữ liệu và nhà phát triển muốn sử dụng Python trong môi trường Android. Như vậy, Kotlin sẽ đảm nhiệm phần interface và Python sẽ thực hiện các thuật toán xử lý ảnh phức tạp. Sự kết hợp này giúp tận dụng điểm mạnh của hai ngôn ngữ để xây dựng ứng dụng di động xử lý ảnh hiệu quả.

## Các thông số ảnh hưởng đến hiệu suất truyền tin

### Tốc độ khung hình

Bộ phát LED có hai trạng thái: BẬT và TẮT từ đó tạo ra các dải tối và sáng trong hình ảnh LED trên cảm biến hình ảnh màn trập lăn. Trong quá trình xử lý hình ảnh, số lượng dải và chiều rộng của dải được sử dụng để khôi phục luồng dữ liệu. Số lượng dải thay đổi tỷ lệ thuận với kích thước của đèn LED.

Tuy nhiên, chiều rộng của dải phụ thuộc chặt chẽ vào tần số BẬT và TẮT của đèn LED. Tần số khác nhau dẫn đến độ rộng dải khác nhau. Chiều rộng của dải sáng và tối có thể là về mặt lý thuyết được tính:

Trong đó biểu thị thời gian cần thiết để đọc ra một dãy(row) pixel của hình ảnh, còn là tần số tương ứng của nguồn sáng LED. [30]

A diagram of a circle with lines and arrows

Description automatically generated

Hình 2‑12: Dải sáng/tối của hệ thống OCC

Một trong những nguyên nhân không thể tránh khỏi tác động đến hiệu suất của hệ thống OCC khi sử dụng camera là khoảng cách giữa các khung hình IFG (Inter-frame gap). Như trong Hình 2‑13(a), thời gian đọc được đo dựa trên khoảng thời gian giữa hai hàng pixel liền kề, và không có khoảng IFG giữa hai khung hình liên tiếp. Trong trường hợp này khung hình tiếp theo sẽ được phơi sáng khi hàng cuối cùng của khung hình trước hoàn thành.

A diagram of a diagram

Description automatically generated

Hình 2‑13: Quá trình xử lý các khung liền kề của cảm biến CMOS  
a) Trường hợp lí tưởng  
b)Trường hợp thực tế

Tuy nhiên, trong thực tế, quá trình tổng hợp dữ liệu của các hàng pixel khác nhau có thể trùng lặp vì mỗi hàng pixel có bộ chuyển đổi tương tự- số ADC (Analog to Digital Converter) riêng, như được chỉ ra trong Hình 2‑13(b). Do đó xuất hiện một khoảng thời gian "trống" giữa hai khung hình liên tiếp mà trong khoảng thời gian này các hàng pixel không được phơi sáng. Khoảng thời gian này được gọi là IFG (, như được minh họa trong Hình 2‑13(b). Khoảng thời gian này là cần thiết để bộ xử lý của camera tổng hợp toàn bộ dữ liệu của một khung, hoặc để chuẩn bị cho khung tiếp theo. [31]

Tốc độ khung hình được xác định:

Với:

* là chiều rộng hình ảnh (pixel),
* là thời gian đọc khung hình
* là khoảng khung.

Khoảng thời gian của một khung hình được chỉ ra trong Hình 2‑14. Thời gian trung bình của một khung là khoảng 33 ms Nó tương ứng với khung tốc độ là 30 khung hình/giây. Thời gian 0.033 ms là bao gồm cả thời gian đọc ( và thời gian IFG . Một số khung hình sẽ mất nhiều thời gian hơn các khung khác, như được chỉ ra ở khung hình 11 và 23 trong kết quả đo đạc của Hình 2‑14. Khoảng thời gian này là do camera dành nhiều thời gian để tổng hợp dữ liệu và do đó làm cho khoảng thời gian cũng dài hơn. Chính nguyên nhân đó làm cho dữ liệu đã truyền đi không được đọc bởi camera, gây khó khăn hơn cho việc khôi phục dữ liệu nếu không có mã sửa lỗi hoặc việc truyền dữ liệu lặp lại. [31]

A graph with blue lines

Description automatically generated

Hình 2‑14: Thời gian cho mỗi khung hình

### Kích thước LED trong ảnh

Như trong Hình 2‑15, kích thước đèn LED thu được được xác đỉnh bởi khoảng cách đèn LED đến máy ảnh (, kích thước thật đèn LED và tiêu cự

Diagram of a sensor with a lens and a diagram

Description automatically generated with medium confidence

Hình 2‑15: Mỗi liên hệ giữ hình ảnh thu được trên cảm biến và hình ảnh thực tế

Từ tính toán ta xác định được kích thước ảnh LED (đường kính)

Với hướng phát triển hệ thống truyền thông định vị trong nhà, yêu cầu truyền 16 bit dữ liệu. Với hệ thống cụm 4 đèn LED, mỗi đèn truyền thông tin của 4 bit dữ liệu.

Từ công thức (1.1) coi (với là kích thước ảnh theo chiều dọc theo đơn vị pixel)

Dữ liệu truyền mỗi bit có kích thước là S (pixel):

(1.4)

Kích thước khung dữ liệu bit trên ảnh được cho bởi:

(1.5)

Từ phương trình (1.3) kích thước ảnh LED (mm) được xác định bởi:

(1.6)

với: là đường kính đèn LED

Kích thước theo pixel của LED:

(1.7)

với : là số cột (pixel) theo độ phân giải hình ảnh thu được, là chiều dọc (mm) theo kích thước cảm biến

A close-up of a screen

Description automatically generated

Hình 2‑16: Ảnh chụp cụm 4 đèn Led ở khoảng cách 1.7m và kích thước của chúng

Thực nghiệm với các thông số : , , ta được theo công thức (1.7) đúng với hình ảnh thu được mô tả ở Hình 2‑16.

Tỉ số nên có thể coi (1.8)

### Cấu trúc khung bản tin

Chiều cao thiết kế của hệ thống LED trong nhà là 3m, tính toán được khoảng cách từ điện thoại đến đèn LED dao động từ . Để thu được toàn bộ khung truyền có kích thước bit và tránh sai số khi xử lí ảnh thì khung truyền cần thoả mãn (1.9)

Từ (1.5), (1.7) và (1.9) ta xác định được số bit truyền tối đa trên mỗi đèn LED:

(1.9)

Có thể thấy : sẽ càng lớn khi

* Tốc độ đọc chậm
* Tốc độ nháy LED nhanh
* Đường kính LED lớn

A screenshot of a phone

Description automatically generated

Hình 2‑17: Thông số camera của 1 thiết bị di động

Áp dụng công thức (1.9) cho hệ thống đang thực nghiệm với bóng đèn LED đường kính , . Với thiết bị như Hình 2‑17 với , hình ảnh thu được có độ phân giải ta được . Do vậy số bit giới hạn trong 1 khung truyền tối đa là 11.

Mã mở đầu có 5 bit với mã bit **“1 0 0 0 1”**. Với điều chế Multi Pulse Position Modulation (MPPM), dữ liệu truyền sau điều chế Để phân biệt vân tối của mã mở đầu với vân tối của bit dữ liệu thì số bit **“0”** tối đa trong phần dữ liệu sau điều chế không quá 2 bit, do đó số bit **“1”** trong phần dữ liệu là . Số bit dữ liệu có thể truyền được tính theo công thức: . Chọn để dữ liệu truyền đạt số bit tối đa.

Vậy ta có các tham số điều chế và

Bảng 2‑7: Cấu trúc khung bản tin

|  |  |
| --- | --- |
| **Preamble** | **Payload**  **(sử dụng điều chế MPPM)** |
| 1 0 0 0 1 | 4B 6B |

Cấu trúc khung bản tin (frame truyền) của Tx được thể hiện trong Bảng 2‑7 bao gồm 5 bit preamble và 6 bit dữ liệu sau khi được điều chế.

# THIẾT KẾ HỆ THỐNG

## Phần phát (Tx)

Ta lựa chọn thiết bị như sau:

* LED tròn đĩa sáng 110cm.
* MOSFET IRF640N
* ESP32 Wrover

A diagram of a power supply system

Description automatically generated

Hình 3‑1: Sơ đồ phần cứng Tx

Sơ đồ nguyên lí sẽ theo hình sau:

A circuit diagram of a device

Description automatically generated

Hình 3‑2: Sơ đồ nguyên lí Tx

### Đèn LED

A close-up of a white and green light

Description automatically generated

Hình 3‑3: Đèn Led âm trần

Đèn LED âm trần là thiết bị chiếu sáng được sử dụng cho rất nhiều công trình, đặc biệt là những nơi có trần thạch cao. Do sự phổ biến của loại đèn LED này trong các công trình nhà ở nên sẽ được lựa chọn trong thí nghiệm này. Thông số chi tiết của đèn LED sử dụng trong đồ án được mô tả trong Bảng 3‑1.

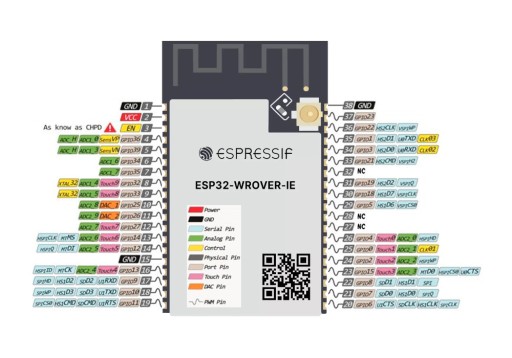
Bảng 3‑1: Thông số đèn LED

|  |  |
| --- | --- |
| Tên | LED Âm Trần 12W Kingled |
| Công suất (W) | 12W |
| Điện áp | 220V/50Hz |
| Quang thông | 1200lm |
| Vùng phát sáng | 110mm |
| Góc chiếu | 120 độ |
| Tiêu Chuẩn | IP 44 |
| Kích thước (ØxH) | (138x50)mm |

### ESP32 WROVER

ESP32-WROVER-IE là một module không dây của dòng vi xử lý ESP32 do Espressif Systems phát triển. Nó tích hợp Wi-Fi và Bluetooth, có khả năng xử lý mạnh mẽ với hai lõi xử lý Tensilica LX6 lên đến 240 MHz. Module này có bộ nhớ flash và RAM tích hợp, hỗ trợ nhiều giao thức truyền thông, GPIO đa dạng cho kết nối linh hoạt, và được sử dụng rộng rãi trong các ứng dụng IoT và các dự án điện tử khác.

Module này có SDK Firmware được cung cấp bởi hãng và rất phù hợp việc cho ứng dụng trong lĩnh vực IoT (Internet of Things) và các dự án điện tử khác như mạng cảm biến không dây, Robotics và IoT di động, sản phẩm kết nối internet, hệ thống điều khiển và tự động hoá,…



Hình 3‑4: Chip điều khiển ESP32 Wrover

Chi tiết thông số phần cứng và thông số phần mềm của ESP32 được mô tả trong Bảng 3‑2 và Bảng 3‑3.

Bảng 3‑2: Thông số phần cứng ESP32.

|  |  |
| --- | --- |
| Điện áp hoạt động | 2.3 ~ 3.6V |
| Hỗ trợ tương tác | Thẻ nhớ, phần cứng UART, SPI, SDIO, I2C, I2S, IR |
| Các GPIO hỗ trợ | Cảm biến chạm, ADC, DAC, bộ tiền khuếch đại (preamplifier) LNA |
| Cảm biến trên chip | Cảm biến Hall, cảm biến nhiệt độ |
| Thạch anh | 40 MHz |
| Dòng hoạt động trung bình | 80 mA |
| Dòng trung bình từ nguồn | 500 mA |
| Nhiệt độ hoạt động | -40°C ~ 85°C |
| Kích thước | 18±0.2 mm x 31.4±0.2 mm x 3.3±0.15 mm |

Bảng 3‑3: Thông số phần mềm ESP32

|  |  |
| --- | --- |
| Hỗ trợ | Cloud Server/ SDK cho phát triển firmware |
| Giao thức | IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT |
| Cấu hình người dùng | AT, cloud server, Android/iOS app |
| WiFi | Hỗ trợ các chế độ: Station, SoftAP, SoftAP và Station, P2P 802.11/g/n/e/i (802.11n tốc độ 150 Mbps). Kết hợp 2 bộ frame truyền A-MPDU và A-MSDU với thời gian chuyển đổi 0.4 µs |
| Tần số | 2.4 GHz ~ 2.5 GHz |
| Bảo mật | WPA/WPA2/WPA2-Enterprise/WPS |
| Mã hóa | AES/RSA/ECC/SHA |
| Nâng cấp firmware | UART download/OTA/download và sửa firmware thông qua host |
| Bluetooth | Bluetooth 4.2 BR/EDR và BLE |

Hoàn thiện hệ thống phần bao gồm 4 đèn LED điều khiển bằng ESP32 ta được như trong Hình 3‑5.

A red circuit board with black wires connected to a group of round black round circular objects

Description automatically generated

Hình 3‑5: Hình ảnh thực tế hệ thống Tx

### Lập trình phần phát

Visual Studio Code (VSCode) là một môi trường phát triển tích hợp (IDE) miễn phí, mã nguồn mở và rất phổ biến được phát triển bởi Microsoft. Đây là một IDE nhẹ, linh hoạt và mạnh mẽ, được thiết kế để hỗ trợ nhiều ngôn ngữ lập trình và môi trường phát triển khác nhau. Quá trình mã hoá dữ liệu, điều khiển nháy LED và cập nhật chương trình được biểu diễn trong Hình 3‑6.

A diagram of a led

Description automatically generated

Hình 3‑6: Lưu đồ mã hoá dữ liệu và điều khiển nháy LED

#### **Mã hoá tín hiệu**

Quá trình mã hoá tín hiệu được thực hiện trong hàm **“void setup()”**:

* Chọn tín hiệu cần truyền ứng với mỗi đèn LED

int binaryNumber1[] = {1, 0, 1, 0}; // Mã cần truyền 1 (10)

int binaryNumber2[] = {1, 0, 0, 1}; // Mã cần truyền 2 (9)

int binaryNumber3[] = {0, 1, 1, 0}; // Mã cần truyền 3 (6)

int binaryNumber4[] = {0, 0, 1, 1}; // Mã cần truyền 4 (3)

* Chuyển dữ liệu từ mã nhị phân sang mã thập phân bằng hàm “binaryToDecimal”:

int binaryToDecimal(int binaryArray[], int size) {

    int decimalValue = 0;

    // Duyệt qua từng bit của mảng nhị phân và tính giá trị thập phân

    for (int i = 0; i < size; ++i) {

        decimalValue = decimalValue \* 2 + binaryArray[i];

    }

    return decimalValue;

}

* Sử dụng **Thuật toán 1** như đã trình bày ở 2.2.2 để má hoá từng tín hiệu với đầu vào là dữ liệu truyền dưới dạng hệ số 10.

// Hàm mã hóa MPPM

int\* encode\_MPPM(int sk, int n, int r) {

    // Khởi tạo khung rỗng

    int\* encode\_MPPM= new int[n];

    while (n > 0) {

        int y;

        if (0 < r && r < n) {

            y = nchoosek(n-1, r);

        } else {

            y = 0;

        }

        // Mã hóa sk bằng cách đặt xung vào các khe có giá trị vị trí phù hợp

        if (y <= sk) {

            sk = sk - y;

            encode\_MPPM[n-1] = 1;

            r = r - 1;

        } else {

            encode\_MPPM[n-1] = 0;

        }

        n = n - 1;

    }

    return encode\_MPPM;

}

* Thêm mã mở đầu **“1 0 0 0 1”** vào dữ liệu sau khi được mã hoá
* ta được khung truyền hoàn chỉnh.
* Điều khiển việc nhấp nháy LED bằng hàm **onTimer().** Ứng với tần số là độ trễ .

My\_timer = timerBegin(0, 80, true);

timerAttachInterrupt(My\_timer, &onTimer, true);

timerAlarmWrite(My\_timer, 50, true);

timerAlarmEnable(My\_timer); //Just Enable

Cụ thể ứng với mỗi bit tín hiệu 1 hoặc 0 đèn LED sẽ bật hoặc tắt, thực hiện lặp lại khi duyệt hết khung tín hiệu.

#### **Quản lí dữ liệu truyền.**

Để có thể nạp firmware từ xa cho ESP32 qua Internet. Chúng ta cần 1 máy chủ lưu giữ. Khi cập nhật hệ thống sẽ tuân theo các bước sau:

• Nhà phát triển Upload firmware lên máy chủ.

• ESP32 sẽ gọi 1 API lên server đó, truy vấn xem có bản cập nhật mới không. Việc hỏi này sẽ được lặp lại liên tục có thể 1 vài phút 1 lần hoặc 1 vài ngày, tùy người phát triển.

• Nếu nhận được thông tin thay đổi version, ESP32 sẽ download firmware về thông qua internet.

• Khởi động lại với firmware mới.

Phần này chúng ta sẽ sử dụng server của OTA Drive. Một website quản lý firmware miễn phí.

A computer hardware components and a computer

Description automatically generated with medium confidence

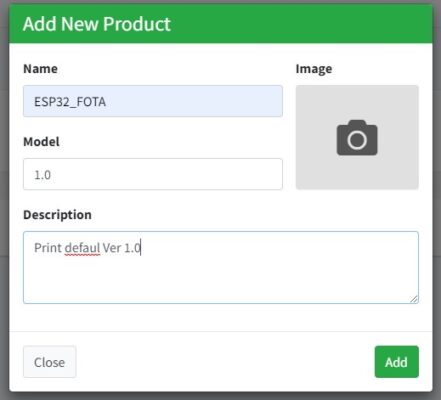
Hình 3‑7: Nguyên lí vận hành kết nối không dây cho ESP32

Các bước nạp firmware không dây cho Esp32 bằng OTA Driver**:**

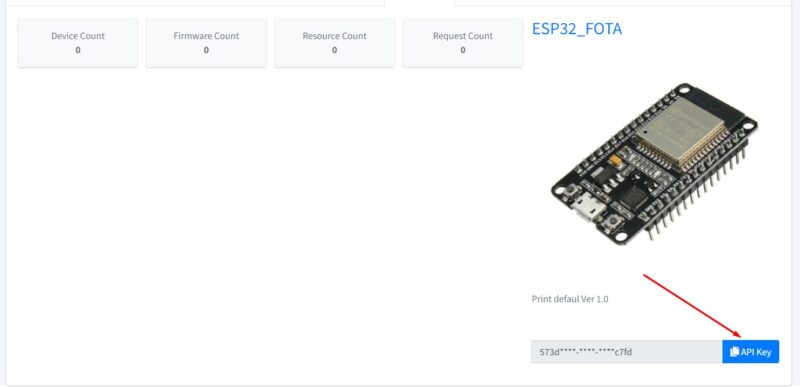
**Bước 1**: Truy cập vào website: https://www.otadrive.com/\

Đăng kí một account và đăng nhập vào

**Bước 2:** Tạo product và lấy mã API của product



Hình 3‑8: Đặt tên sản phẩm và version mặc định



Hình 3‑9: Copy API key

**Bước 3:** Lập trình cho ESP32:

Lập trình ESP32 FOTA Internet:

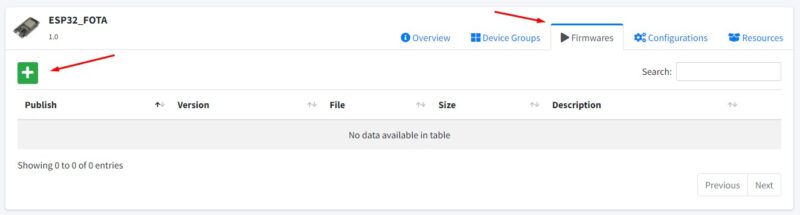
* Hàm cập nhật qua OTA
* void update\_FOTA()
* {
* String url = "http://otadrive.com/deviceapi/update?";
* url += "k=" + key;
* url += "&v=" + version;
* url += "&s=" + getChipId(); // định danh thiết bị trên Cloud
* WiFiClient client;
* httpUpdate.update(client, url, version);
* }

Trong hàm này chúng ta gọi API tới OTA Drive với các tham số truyền vào là:

* K: Key API chúng ta đã lấy ở bước trước
* V: Version firmware hiện tại để so sánh với version lưu trữ trên server
* S: Là ID của chip, ID này là duy nhất với mỗi con ESP32 để định danh chúng với các ESP32 khác dùng chung firmware.
* Sử dụng hàm getID để lấy ID của ESP32
* String getChipId()
* {
* String ChipIdHex = String((uint32\_t)(ESP.getEfuseMac() >> 32), HEX);
* ChipIdHex += String((uint32\_t)ESP.getEfuseMac(), HEX);
* return ChipIdHex;
* }
* Cuối cùng, chúng ta sẽ gọi hàm update mỗi 1phút 1 lần ( 120 lần count 500ms) trong hàm **“void loop()”**

**Bước 4:** Verify thiết bị trên Server để ESP có thể truy cập vào server.

**Bước 5:** Upload firmware lên OTA Drive:



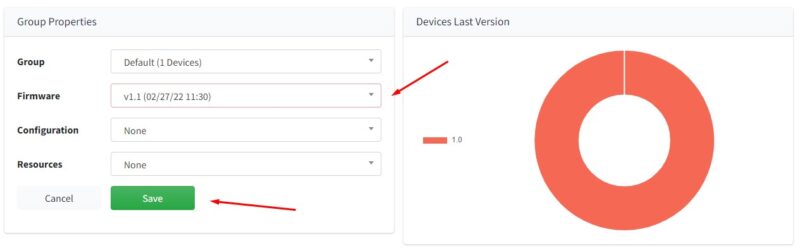
Hình 3‑10: Mở tab firmware và ấn add new

A screenshot of a computer

Description automatically generated

Hình 3‑11: Upload các phiên bản của bạn

Thay đổi version firmware và FOTA Internet vào ESP32.



Hình 3‑12: Chọn phiên bản cần sử dụng, sau đó nhấn Save

Sau khi thay đổi version firmware, hệ thống sẽ tự động cập nhật chương trình mới trong 3-5p và sẽ kiểm tra phiên bản mỗi 1 phút ( có thể thay đổi bộ hẹn giờ cập nhật trong chương trình).

### Kết quả triển khai phần phát

A screenshot of a computer screen

Description automatically generated

Hình 3‑13: Hình vẽ Frame truyền chi tiết

Sau khi lập trình xong phần điều khiển Led, đợi 1-2s sau khi hệ thống Setup xong cụm đèn Led thu được sau khi chụp bằng thiết bị di động sẽ như Hình 2‑13.

Ví dụ với LED 1 mang dữ liệu là “1, 0, 0, 1” sau khi mã hoá MPPM(6,4) thu được dãy bit “1, 1, 0, 0, 1, 1”, kết hợp với **Preamble** “1, 0, 0 ,0 ,1” ta được frame truyền là “1, 0, 0 ,0 ,1, 1, 1, 0, 0, 1, 1”. Frame truyền được đánh dấu bắt đầu và kết thúc bởi 2 vân tối lớn nhất.

## Phần thu (Rx)

Hệ thống Rx là một phần mềm trên thiết bị di động được xây dựng trên nền tảng Android Studio với biểu đồ tuần tự được thể hiện trên Hình 3‑14.

A diagram of a software development process

Description automatically generated

Hình 3‑14: Biểu đồ tuần tự của phần mềm

### Thiết bị sử dụng cho phần thu

|  |  |
| --- | --- |
| **Tên thông số** | **Giá trị** |
| Hãng điện thoại | Vsmart Aris |
| CPU | Qualcomm Snapdragon 730 8 nhân KryoTM 470 (tối đa 2.2 GHz) |
| GPU | 257MHz |
| Cảm biến ảnh | Rolling Shutter CMOS |
| Tốc độ khung hình | 30 fps |
| Camera trước | 20 MP f2.2 |
| Camera sau | 64MP F1.89 camera chụp đêm  8MP F2.2 camera góc rộng  8MP F2.4 camera chụp xa 2x  2MP F2.4 camera macro |
| Tiêu cự | Camera trước: 2.82mm  Camera sau: 5.43mm |
| Camera API | Camera2 API level 30 |
| Độ phân giải hiển thị | Full HD+ (1080 x 2340 Pixels) |

### Thuật toán xử lí ảnh

A diagram of a process

Description automatically generated

Hình 3‑15: Sơ đồ khối quá trình xử lí ảnh

Cụ thể quá trình xử lí có các phần chính như sau:

* Chuyển đổi ảnh dưới định dạng bitmap (BMP) sang mảng Numpy:
* img = np.frombuffer(frame, dtype=np.uint8).reshape(heigth, width, 4)
* Chuyển sang ảnh xám:
  + image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
  + image = cv2.convertScaleAbs(image, alpha=255.0/np.max(image))

|  |  |
| --- | --- |
| A group of circular lights  Description automatically generated  Hình 3‑16a: Ảnh màu | A group of white circles on a black background  Description automatically generated  Hình 3‑17b: Ảnh xám |

* Chuyển ảnh xám sang ảnh nhị phân, tìm contours:

Thực hiện chuyển ảnh xám qua ảnh nhị phân bằng hàm sau:

* \_, threshold\_image = cv2.threshold(image, threshold\_read, 255, cv2.THRESH\_BINARY  )

Trước đó đã đưa ảnh về thang độ xám tối đa là 255 nên ta chọn thresholdread=200.

Sau đó ta áp dụng thuật toán tìm contours. Trong xử lý ảnh và computer vision, contours (đường viền) là các đường cong liên tục đại diện cho các ranh giới của các đối tượng trong ảnh.

Trong OpenCV, ta có thể sử dụng hàm cv2.findContours() để tìm contours trong ảnh. Hàm này trả về danh sách các contours, mỗi contour được biểu diễn dưới dạng một danh sách các điểm (x, y).

* + contours, \_ = cv2.findContours(threshold\_image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

Sau đó vẽ một hình chữ nhật bao quanh mỗi contours liền nhau bằng hàm **‘boundingRect’**. Hình chữ nhật này thường được gọi là "bounding rectangle" hoặc "bounding box".

* + x, y, w, h = cv2.boundingRect(contour)
  + # Lưu trữ bounding rectangle
  + bounding\_boxes.append((x, y, w, h))

Bounding rectangle được đặc trưng bởi tọa độ (x, y) của góc trên bên trái và chiều rộng (width) cùng chiều cao (height) của hình chữ nhật bao quanh các contours.

A group of white circles on a black background

Description automatically generated

Hình 3‑18: Ảnh nhị phân đã vẽ contours

* Ghép các **boundingRect** gần nhau thành các cụm (LED):

Vì các **boundingRect** được phát hiện ngẫu nhiên nên chưa thể tiến hành đọc dữ liệu ngay, cần tiến ghép chúng thành các cụm (LED).

Đầu tiên ta quy ước các vị trí "bounding box" là tâm của chúng với toạ độ (x,y) lưu trữ vào 1 list. Tiếp theo sử dụng DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán phân cụm dựa trên mật độ giúp xác định tất cả các điểm trong bán kính ε (epsilon) từ điểm đã chọn.

Theo công thức (1.8), các đèn LED được thiết kế cách nhau tối thiểu 0.1m, khoảng cách từ đèn LED tới thiết bị tối đa 2m, ta có được khoảng cách tối thiểu các đèn Led trong ảnh cho bởi:

(pixel)

Với width là chiều rộng của ảnh.

Ta có hàm DBSCAN với các thông số như sau:

* + dbscan = DBSCAN(eps=round(width\*0.1/2), min\_samples=2)
  + labels = dbscan.fit\_predict(center\_array)

Trong đó:

* **Epsilon (ε):** Bán kính xác định vùng lân cận của mỗi điểm.
* **MinPts:** Số lượng điểm tối thiểu cần phải có trong mỗi vùng lân cận để một điểm được coi là trung tâm của một cụm.

Từ đây các vân sáng sẽ được ghép lại theo các vị trí LED tương ứng và được tiến hành đọc dữ liệu theo các đèn.

* Tiến hành đọc dữ liệu từng LED

Quy trình thực hiện việc đọc dữ liệu được mô tả trong Hình 3‑19.

A diagram of a flowchart

Description automatically generated

Hình 3‑19: Lưu đồ thuật toán đọc dữ liệu trên LED

Sau khi có được toạ độ các vân sáng, sử dụng vòng lặp để tính độ rộng các vân tối theo công thức:

* + dark\_size = [bounding\_boxes[i+1][0] - bounding\_boxes[i][0] - bounding\_boxes[i][2] for i in range(len(bounding\_boxes)-1)]
  + long\_size = max(dark\_size)

Sử dụng hàm max để tìm các vân tối rộng nhất.(3bit)

Từ khoảng cách 2 vân tối rộng nhất liền kề, ta có được độ rộng 1 frame dữ liệu (11 bit) bao gồm 5 bit preamble và 6 bit payload mã hoá MPPM. Lấy trung bình ta được độ rộng 1 bit dữ liệu .

Từ độ rộng 1 bit dữ liệu, ta duyệt qua tất cả vân sáng và tối, số bit 1 tương ứng với , bit 0 tương ứng với . Tổng hợp lại được frame dữ liệu hoàn chỉnh. Bỏ qua pramble “1, 0, 0, 1” ta được dữ liệu sau điều chế bằng mã MPPM.

A white rectangular object with a red arrow

Description automatically generated

Hình 3‑20: Toạ độ các bounding box

* Sau khi có được frame dữ liệu dữ liệu của từng LED, ta tiến hành giải mã bằng **Thuật toán 2** :

|  |
| --- |
| **Thuật toán 2:** Thuật toán giải điều chế |
| **Input:** Codeword có độ dài  **Output:** Kí tự dữ liệu  **For** **to**  **do**  **| if then**  **| | if**  **then**  **| | |\_**  **| |**\_  **|**\_ |

Dữ liệu trả về sẽ được lưu lại và xem xét khi nhận đủ dữ liệu 5 lần của 1 đèn, dữ liệu xuất hiện nhiều nhất sẽ được lưu trữ và hiển thị.

Để phục vụ đánh giá hệ thống sẽ tiến hành lặp lại toàn bộ quá trình trên 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.

# KẾT QUẢ THỰC NGHIỆM

## Triển khai thực nghiệm

Tiến hành thực nghiệm trên thiết bị với các tần số khác nhau để đo tỉ lệ khung hình lỗi và thời gian xử lí mỗi hình.

|  |  |
| --- | --- |
| A screenshot of a phone  Description automatically generated  Hình 4‑1a: Kết quả chạy trên app Android | Hình 4‑2b: Thông số thiết bị (camera trước) |

Sử dụng tính năng “Logcat” trong phần mềm Android Studio để ghi lại kết quả xử lí của 100 khung hình, với kết quả trả về là dữ liệu trống chính là những khung lỗi không thể giải mã. **Tỉ lệ khung hình có thể giải mã** là số khung hình có thể giải mã trên tổng số khung hình truyền sẽ được khảo sát và đưa lên đồ thị.

Bên cạnh đó **tỉ lệ dữ liệu lỗi** cũng được khảo sát và đánh giá. Tỉ lệ dữ liệu lỗi được tính bằng cách lấy trung bình dữ liệu giải mã sai trên tổng số khung hình có thể giải mã của 4 đèn LED.

### Thực nghiệm với tần số 14kHz

Đặt tham số thời gian trong hàm ngắt tương ứng với tần số .

Bảng 4‑1: Kết quả thực nghiệm với tần số 14kHZ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Khoảng cách (m) | Tỉ lệ khung hình có thể giải mã (%) | | | | Tỉ lệ dữ liệu lỗi (%) |
| LED 1 | LED 2 | LED 3 | LED 4 |
| 1.5 | 90 | 91 | 96 | 91 | 0 |
| 1.6 | 81 | 84 | 84 | 81 | 0 |
| 1.7 | 71 | 70 | 73 | 71 | 0 |
| 1.8 | 62 | 56 | 65 | 54 | 0 |
| 1.9 | 61 | 50 | 61 | 55 | 0 |
| 2.0 | 44 | 39 | 42 | 39 | 0 |
| 2.1 | 38 | 35 | 34 | 35 | 0 |
| 2.2 | 30 | 28 | 21 | 22 | 0 |
| 2.3 | 0 | 0 | 0 | 0 | 0 |
| 2.4 | 0 | 0 | 0 | 0 | 0 |
| Thời gian xử lí (s) | 0.0082 | | | |  |

### Thực nghiệm với tần số 18kHZ

Đặt tham số thời gian trong hàm ngắt tương ứng với tần số .

Bảng 4‑2: Kết quả thực nghiệm với tần số 18kHZ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Khoảng cách (m) | Tỉ lệ khung hình có thể giải mã (%) | | | | Tỉ lệ dữ liệu lỗi (%) |
| LED 1 | LED 2 | LED 3 | LED 4 |
| 1.5 | 99 | 98 | 99 | 97 | 0 |
| 1.6 | 98 | 98 | 99 | 97 | 0 |
| 1.7 | 99 | 97 | 98 | 93 | 0.25 |
| 1.8 | 96 | 93 | 97 | 90 | 0.28 |
| 1.9 | 91 | 91 | 98 | 89 | 0 |
| 2.0 | 91 | 91 | 96 | 84 | 0 |
| 2.1 | 82 | 79 | 80 | 80 | 0 |
| 2.2 | 75 | 77 | 75 | 73 | 0 |
| 2.3 | 74 | 62 | 71 | 66 | 0 |
| 2.4 | 69 | 57 | 65 | 65 | 0 |
| Thời gian xử lí (s) | 0.0145 | | | | |

### Thực nghiệm với tần số 20kHz

Đặt tham số thời gian trong hàm ngắt tương ứng với tần số .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Khoảng cách (m) | Tỉ lệ khung hình có thể giải mã (%) | | | | Tỉ lệ dữ liệu lỗi (%) |
| LED 1 | LED 2 | LED 3 | LED 4 |
| 1.5 | 99 | 96 | 96 | 98 | 0.25 |
| 1.6 | 98 | 95 | 96 | 93 | 0.26 |
| 1.7 | 96 | 94 | 93 | 91 | 0.27 |
| 1.8 | 94 | 94 | 93 | 90 | 1.05 |
| 1.9 | 88 | 94 | 92 | 89 | 1.42 |
| 2.0 | 87 | 92 | 92 | 85 | 1.42 |
| 2.1 | 79 | 82 | 90 | 75 | 3.33 |
| 2.2 | 72 | 76 | 87 | 74 | 3.98 |
| 2.3 | 70 | 75 | 86 | 70 | 4.01 |
| 2.4 | 67 | 61 | 76 | 63 | 4.99 |
| Thời gian xử lí (s) | 0.0150 | | | | |

Bảng 4‑3: Kết quả thực nghiệm với tần số 20kHZ

## Đánh giá kết quả đạt được

### Tần số 14kHz

Với thực nghiệm số 1 ta có biểu đồ đường như sau:

A graph of a graph with different colored lines

Description automatically generated with medium confidence

Hình 4‑3: Đồ thị đường trong thực nghiệm với tần số 14kHz

Trong khoảng cách dưới độ tỉ lệ khung hình có thể giải mã trên bởi với tần số 14kHz thì khung dữ liệu vẫn đảm bảo nằm trong ảnh đèn LED. Từ đến dữ liệu giảm dần xuống dưới và giảm mạnh sau bởi lúc này kích thước đèn LED còn khá nhỏ, khung dữ liệu không đảm bảo luôn chứa đủ trong ảnh đèn LED khiến việc đọc dữ liệu khá khó khăn. Bên cạnh đó việc giải mã dữ liệu lại luôn đem lại kết quả chính xác bởi kích thước các vân khá lớn.

Có thể thấy tần số 14kHz không tối ưu trong khoảng cách ngoài 1.9m với tỉ lệ khung hình lỗi cao nhưng bù lại khả năng giải mã dữ liệu mạnh với độ chính xác tuyệt đối và thời gian xử lí hình ảnh nhanh với thời gian là .

### Tần số 18kHz

Với thực nghiệm số 2 ta có biểu đồ đường như sau:

A graph of different colored lines

Description automatically generated

Hình 4‑4: Đồ thị đường trong thực nghiệm với tần số 18kHz

Trong thực nghiệm với tần số , tỉ lệ khung hình chính xác luôn lớn hơn trong phạm vi . Từ tỉ lệ khung hình có thể đọc được giảm nhưng không đáng kể vẫn trong phạm vi tối ưu cho việc giải mã và có xu hướng giảm mạnh khi ra xa hơn khi tỉ lệ khung hình có thể giải mã trung bình chỉ còn 60% ở khoảng cách 2.4m. Tỉ lệ giải mã sai dữ liệu luôn không quá 2.8% trong phạm vi .

Tuy tỉ lệ chính xác cao nhưng bù lại thì thời gian xử lí là , lâu hơn so với tần số 14kHz nhưng vẫn đảm bảo khung hình ổn định, giảm thiểu độ trễ khi nhận và xử lí dữ liệu với tỉ lệ chính xác rất cao trong phạm vi thiết – phạm vi thiết kế tối đa của hệ thống định vị trong nhà.

### Tần số 20kHz

Với thực nghiệm số 3 ta có biểu đồ đường như sau:

A graph of different colored lines

Description automatically generated

Hình 4‑5: Đồ thị đường trong thực nghiệm với tần số 20kHz

Khi tăng tần số LED lên 20kHz, nhìn chung tỉ lệ khung hình lỗi vẫn rất thấp so với tần số 14kHz. Khả năng giải mã luôn đảm bảo lớn hơn trong phạm vi . Tuy nhiên ngoài thì tỉ lệ dữ liệu lỗi lại cao hơn tần số 18kHz bởi tần số lớn khiến kích thước pixel của 1 bit dữ liệu nhỏ hơn, chịu ảnh hưởng bởi nhiễu môi trường khiến các vân tối bị nhoè và khó khăn cho việc giải mã. Vì thế tỉ lệ giải mã sai dữ liệu có xu hướng tăng mạnh, lên đến 5% ở khoảng cách .

Thời gian xử lí là cao hơn so với tần số 18kHz bởi việc tăng tần số đồng nghĩa với tăng số bit cần xử kí trong 1 khung hình. Cùng với đó tỉ lệ dữ liệu lỗi tăng mạnh cho thấy tần số không tối ưu bằng tần số .

# KẾT LUẬN

## Kết luận

Qua quá trình nghiên cứu và hoàn thành đồ án tốt nghiệp với đề tài **“Nghiên cứu phương pháp mã hoá mới nhằm nâng cao khả năng truyền dữ liệu cho truyền thông bằng ánh sáng khả kiến”** em có được những kiến thức về:

* Tìm hiểu tổng quan về công nghệ giao tiếp không dây quang học (OWC) đặc biệt là giao tiếp camera quang học (OCC) ứng dụng định vị trong nhà
* Lập trình C/C++ kiểm soát hệ thống đèn Led để phát tín hiệu
* 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:

* Lập trình cho Chip điều khiển ESP32 Wrover với nhiệm vụ: kiểm soát tần số nháy LED theo dữ liệu vị trí đã được mã hoá bằng phương pháp MPPM, quản lí dữ liệu truyền trên OTA driver
* Triển khai thành công hệ thống điều chế MPPM giúp tăng kích thước dữ liệu truyền vẫn đảm bảo không xảy ra hiện tượng nhấp nháy cho mắt người
* Tối ưu hoá phần mềm cung cấp hình ảnh và trả kết quả về giao diện người dùng chạy trên hệ điều hành Android
* Đề xuất thuật toán xử lí ảnh giúp phát hiện RoI và tăng tốc đọc dữ liệu từ hình ảnh thu được với độ chính xác cao
* Giải mã dữ liệu và trả về dữ liệu từng đèn LED
* Tiến hành thực nghiệm và đánh giá kết quả.

## Hướng phát triển

Với sự phát triển ngày càng mạnh mẽ của công nghệ camera 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 hệ thống phát giúp quản lí dữ liệu linh hoạt hơn
* Tối ưu hoá phần mềm, tăng tốc độ xử lí dữ liệu và nâng cao độ chính xác, dễ tiếp cận hơn với người sử dụng
* Xây dựng thuật toán định vị từ dữ liệu thu được của từng đèn LED

Định vị trong nhà đang là một nhu cầu cần thiết ở hiện tại và tương lai. Việc áp dụng OCC 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] | Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," Cisco: San Jose, CA, USA, 2020. |
| [2] | X. Wang, L. Kong, F. Kong, F. Qiu, M. Xia, S. Arnon and G. Chen, "Millimeter wave communication: A comprehensive survey," 2018, p. 1616–1653. |
| [3] | Y. Niu, Y. Li, D. Jin, L. Su và A. Vasilakos, “A survey of millimeter wave communications (mmWave) for 5G: Opportunities and challenges,” p. 2657–2676, 2015. |
| [4] | N. Chi, Y. Zhou, Y. Wei and F. Hu, "Visible light communication in 6G: Advances, challenges, and prospects.," 2020, pp. 15, 93–102. |
| [5] | W. Liu and Z. Xu, "Some practical constraints and solutions for optical camera communication.," 2020. |
| [6] | J. Wang, J. Liu, S. Li, Y. Zhao, J. Du and L. Zhu, "Orbital angular momentum and beyond in free-space optical communications," 2021. |
| [7] | "Using a CMOS camera sensor for visible light communication," In Proceedings of the 2012 IEEE Globecom Workshops, Anaheim, CA, USA, 3–7 December 2012. |
| [8] | H. Lee, H. Lin, Y. Wei, H. Wu, H. Tsai and K. Lin, "Rollinglight: Enabling line-of-sight light-to-camera communications," In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, Florence, Italy, 20–22 May 2015. |
| [9] | P. Luo, Z. Ghassemlooy, H. Le Minh, X. Tang and H. Tsai, "Undersampled phase shift ON-OFF keying for camera communication," In Proceedings of the 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), Hefei, China, October 2014. |
| [10] | H. Chen, S. Wen, X. Wang, M. Liang, M. Li, Q. Li and Y. Liu, "Color-shift keying for optical camera communication using a rolling shutter mode," IEEE Photonics J., 2019. |
| [11] | V. Rachim and W. Chung, "Multilevel intensity-modulation for rolling shutter-based optical camera communication," IEEE Photonics Technol. Lett. , 2018. |
| [12] | Y. Kuo, P. Pannuto, K. Hsiao and P. L. Dutta, " Indoor positioning with mobile phones and visible light," In Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, Maui, HI, USA, September 2014. |
| [13] | "Basic knowledge of LED - White LED," http://3fire.blogspot.com/2008/06/led-led.html. |
| [14] | A. El Gamal and H. Eltoukhy, "CMOS image sensors.," in *IEEE Circuits Devices Mag*, 2005, pp. 21, 6–20. |
| [15] | A. El Gamal and H. Eltoukhy, "CMOS image sensors," IEEE Circuits Devices Mag. , 2005. |
| [16] | Z. Chen, X. Wang, S. Pacheco and R. Liang, " Impact of CCD camera SNR on polarimetric accuracy," Appl. Opt. 2014, pp. 53, 7649–7656. |
| [17] | H. Yang, W. Zhong, C. Chen and Alphones, " A. Integration of visible light communication and positioning within 5G networks for internet of things," in *IEEE Netw*, 2020, pp. 34, 134–140. |
| [18] | P. Lin, X. Hu, Y. Ruan, H. Li, J. Fang, Y. Zhong, H. Zheng, J. Fang, Z. Jiang and Z. Chen, "Real-time visible light positioning supporting fast moving speed," in *Opt. Express*, 2020, pp. 28, 14503–14510. |
| [19] | Y. Li, Z. Ghassemlooy, X. Tang, B. Lin and Y. Zhang, "A VLC smartphone camera based indoor positioning system," in *EEE Photonics Technol. Lett*, 2018, pp. 30, 1171–1174. |
| [20] | M. Hasan, M. Ali, M. Rahman, M. Chowdhury and Y. Jang, "Optical camera communication in vehicular applications:," A review. IEEE Trans. Intell. Transp. Syst, 2021. |
| [21] | P. Ji, H. Tsai, C. Wang and F. Liu, "Vehicular visible light communications with LED taillight and rolling shutter camera," in *In Proceedings of the 2014 IEEE 79th Vehicular Technology Conference (VTC Spring),*, Seoul, Republic of Korea, 18–21 May 2014, p. 1–6. |
| [22] | K. Xu, K. Zhou, C. Zhu, S. Zhang, B. Shi, X. Li, T. Huang and C. Xu, "When Visible Light (Backscatter) Communication Meets Neuromorphic Cameras in V2X," in *In Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications*, CA, USA, 22–23 February 2023, p. 42–48. |
| [23] | N. Van Hoa, H. Nguyen, C. Nguyen and Y. Jang, "OCC Technology-based Developing IoT Network," In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, October 2020. |
| [24] | M. Z. C. M. S. V. T. N. a. Moh. Khalid Hasan, "Performance Analysis and". |
| [25] | M. Vasilakis, "DynaLight: A Dynamic Visible Light Communication Link for Smartphones," Master’s Thesis in Embedded Systems. |
| [26] | Y. Kozawa and H. Habuchi, "Enhancement of optical wireless multi pulse PPM," IEEE Global Commun, 2008. |
| [27] | M. K. a. S. B. F. Xu, "Coded PPM and multipulse PPM and iterative detection for free-space optical links," IEEE/OSA J.Optical Commun. Netw, 2009. |
| [28] | A. B. S. a. M. Tahir, "Joint Rate-Brightness Control using Variable Rate MPPM for LED Based Visible Light Communication Systems," Member, IEEE. |
| [29] | K. N. H. Sugiyama, Journal of Lightwave Technology, pp. 465 - 472. |
| [30] | T.-H. Do, "Analysis on Visible Light Communication using Rolling Shutter CMOS Sensor," University of Information Technology VNU-HCM, Ho Chi Minh City, Vietnam. |
| [31] | P. V. K. Nguyễn Duy Thông, "PHÂN TÍCH CÁC THÔNG SỐ ẢNH HƯỞNG ĐẾN HIỆU SUẤT CỦA HỆ THỐNG TRUYỀN THÔNG QUANG SỬ DỤNG CAMERA," Trường Đại học Quy Nhơn, Việt Nam. |
| [32] | M. Chowdhury, M. Shahjalal, M. Hasan and Y. Jang, "The role of optical wireless communication technologies in 5G/6G and IoT solutions: Prospects, directions, and challenges.," 2019. |
| [33] | H. Chen, C. Wu, H. Li, X. Chen, Z. Gao, S. Cui and Q. Wang, "Advances and prospects in visible light communications," 2016. |
| [34] | Q. Shan, C. Wei, Y. Jiang, J. Song, Y. Zou, L. Xu, T. Fang, T. Wang, Y. Dong and J. Liu, "Perovskite light-emitting/detecting bifunctional fibres for wearable LiFi communication," 2020. |
| [35] | B. Lin, Z. Ghassemlooy, C. Lin, X. Tang, Y. Li and S. Zhang, "An indoor visible light positioning system based on optical camera communications," IEEE Photonics Technol. Lett, 2017. |

# PHỤ LỤC

* **Chương trình điều khiển phần phát tín hiệu:**

#include <Arduino.h>

#include <HTTPUpdate.h>

#include <WiFi.h>

#include <WiFiMulti.h>

#define LED1 27

#define LED2 14

#define LED3 33

#define LED4 32

// Biến khởi tạo

WiFiMulti wifiMulti;

String version = "2.2";

hw\_timer\_t \*My\_timer = NULL;

uint32\_t updateCounter = 0;

uint32\_t counter = 0;

const uint32\_t connectTimeoutMs = 10000;

String key = "e3f99ca6-cc4a-48a1-b3ce-aad15db116ff";

int n = 6;

int r = 4;

int arrayHeader[]= { 1, 0, 0, 0, 1}; // Mã header

int sizeHeader = sizeof(arrayHeader)/ sizeof(arrayHeader[0]);

int binaryNumber1[] = {1, 0, 1, 0}; // Mã cần truyền 1 (10)

int binaryNumber2[] = {1, 0, 0, 1}; // Mã cần truyền 2 (9)

int binaryNumber3[] = {0, 1, 1, 0}; // Mã cần truyền 3 (6)

int binaryNumber4[] = {0, 0, 1, 1}; // Mã cần truyền 4 (3)

int sizeNumber = sizeof(binaryNumber1)/sizeof(binaryNumber1[0]);

int size = n+sizeHeader;

int\* code1 = new int[size];

int\* code2 = new int[size];

int\* code3 = new int[size];

int\* code4 = new int[size];

int i = 0;

// Hàm tính tổ hợp C(n, k)

unsigned long long nchoosek(int n, int k) {

    if (k == 0 || k == n) {

        return 1;

    }

    if (k > n - k) {

        k = n - k;

    }

    unsigned long long result = 1;

    for (int i = 1; i <= k; ++i) {

        result \*= n - i + 1;

        result /= i;

    }

 return result;

}

// Hàm gán giá trị của một mảng cho một mảng khác

void assignArray(int source[], int destination[], int size) {

    for (int i = 0; i < size; ++i) {

        destination[i] = source[i];

    }

}

// Hàm thêm giá trị của một mảng vào cuối một mảng khác

void appendArrays(int source[], int destination[], int sizeSource, int sizeDestination) {

    for (int i = 0; i < sizeSource; ++i) {

        destination[sizeDestination + i] = source[i];

    }

}

// Hàm mã hóa MPPM

int\* encode\_MPPM(int sk, int n, int r) {

    // Khởi tạo khung rỗng

    int\* encode\_MPPM= new int[n];

    while (n > 0) {

        int y;

        if (0 < r && r < n) {

            y = nchoosek(n-1, r);

        } else {

            y = 0;

        }

        // Mã hóa sk bằng cách đặt xung vào các khe có giá trị vị trí phù hợp

        if (y <= sk) {

            sk = sk - y;

            encode\_MPPM[n-1] = 1;

            r = r - 1;

        } else {

            encode\_MPPM[n-1] = 0;

        }

        n = n - 1;

    }

    return encode\_MPPM;

}

// Hàm chuyển mảng nhị phân sang thập phân

int binaryToDecimal(int binaryArray[], int size) {

    int decimalValue = 0;

    // Duyệt qua từng bit của mảng nhị phân và tính giá trị thập phân

    for (int i = 0; i < size; ++i) {

        decimalValue = decimalValue \* 2 + binaryArray[i];

    }

    return decimalValue;

}

void IRAM\_ATTR onTimer(){

  if(i >= size){i = 0;}

  digitalWrite(LED1, code1[i]);

  digitalWrite(LED2, code2[i]);

  digitalWrite(LED3, code3[i]);

  digitalWrite(LED4, code4[i]);

  i++;

}

void update\_FOTA();

void setup() {

  Serial.begin(9600);  // Khởi tạo kết nối với cổng Serial

  WiFi.mode(WIFI\_STA);

  // Thêm danh sách wifi

  wifiMulti.addAP("IoTEAM\_Lab\_VLC", "A11111z22222");

  // Kết nối với Wi-Fi sử dụng wifiMulti (kết nối với SSID có tín hiệu mạnh nhất)

  Serial.println("Connecting Wifi...");

  if(wifiMulti.run() == WL\_CONNECTED) {

    Serial.println("");

  }

  int dataPackage[size]; // Gói tin dùng chung

  assignArray(arrayHeader, dataPackage, sizeHeader);

  // Truyền gói tin vào các LED 1

  int decimalNumber = binaryToDecimal(binaryNumber1,sizeNumber);

  int\* encodedFrame = encode\_MPPM(decimalNumber, n, r);

  appendArrays(encodedFrame, dataPackage, n, sizeHeader );

  delete [] encodedFrame;

  assignArray(dataPackage, code1, size);

  // Truyền gói tin vào các LED 2

  decimalNumber = binaryToDecimal(binaryNumber2,sizeNumber);

  encodedFrame = encode\_MPPM(decimalNumber, n, r);

  appendArrays(encodedFrame, dataPackage, n, sizeHeader );

  delete [] encodedFrame;

  assignArray(dataPackage, code2, size);

  // Truyền gói tin vào các LED 3

  decimalNumber = binaryToDecimal(binaryNumber3,sizeNumber);

  encodedFrame = encode\_MPPM(decimalNumber, n, r);

  appendArrays(encodedFrame, dataPackage, n, sizeHeader );

  delete [] encodedFrame;

  assignArray(dataPackage, code3, size);

  // Truyền gói tin vào các LED 4

  decimalNumber = binaryToDecimal(binaryNumber4,sizeNumber);

  encodedFrame = encode\_MPPM(decimalNumber, n, r);

  appendArrays(encodedFrame, dataPackage, n, sizeHeader );

  delete [] encodedFrame;

  assignArray(dataPackage, code4, size);

  // Cài đặt ngõ đầu ra tín hiệu LED

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

  My\_timer = timerBegin(0, 80, true);

  timerAttachInterrupt(My\_timer, &onTimer, true);

  timerAlarmWrite(My\_timer, 58, true); // 500 microseconds= 0.5ms, t=1/f f=2000

  timerAlarmEnable(My\_timer); //Just Enable

}

void loop()

{

  //Nếu mất kết nối với điểm phát sóng mạnh nhất, nó sẽ kết nối với mạng tiếp theo trong danh sách

  if (wifiMulti.run(connectTimeoutMs) == WL\_CONNECTED) {

  }

  delay(500);

  if (WiFi.status() == WL\_CONNECTED)

  {

    updateCounter++;

    if (updateCounter > 120)

    {

      updateCounter = 0;

      update\_FOTA();

    }

  }

}

String getChipId()

{

  String ChipIdHex = String((uint32\_t)(ESP.getEfuseMac() >> 32), HEX);

  ChipIdHex += String((uint32\_t)ESP.getEfuseMac(), HEX);

  return ChipIdHex;

}

void update\_FOTA()

{

  String url = "http://otadrive.com/deviceapi/update?";

  url += "k=" + key;

  url += "&v=" + version;

  url += "&s=" + getChipId(); // định danh thiết bị trên Cloud

  WiFiClient client;

  httpUpdate.update(client, url, version);

}

* **Chương trình python xử lí ảnh:**

import numpy as np

import cv2

import math

import time

from com.ledid.lediddection.model import Data

from sklearn.cluster import DBSCAN

# khai báo tham số

thresholdCount = 100 # ngưỡng xám để lọc pixel có giá trị cao

length\_data = 6 # độ dãi dữ liệu đã điều chế

number\_one = 4 # số bit 1 trong dữ liệu đã điều chế

length\_header = 5 # độ giải mã mở đầu

nDefault = 18

def main(frame,heigth, width,threshold,nLED,camera):

    try:

        start\_time = time.time()

        # đọc ảnh từ tập tin

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

        image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

        image = cv2.convertScaleAbs(image, alpha=255.0/np.max(image))

        # goị hàm trích xuất RoI

        data\_list, rois = grid\_division(image, nLED,threshold,width,camera)

        x=[]

        y=[]

        w=[]

        h=[]

        check = 0

        list\_code = ["" for \_ in range(4)]

        # giải mã dữ liệu và in kết quả

        if len(rois) == nLED:

            check = 1

            for i, data in enumerate (data\_list):

                x.append(rois[i][0])

                y.append(rois[i][1])

                w.append(rois[i][2])

                h.append(rois[i][3])

                if data:

                    list\_code[i] = decoder(data)

        end\_time = time.time()

        # Tính thời gian xử lý

        elapsed\_time = end\_time - start\_time

        return Data(elapsed\_time, w, h, x, y, list\_code[0], list\_code[1], list\_code[2], list\_code[3],check )

    except:

        return Data(0.0,[], [], [] ,[],  str(""),str(""),str(""), str(""), 0)

# định nghĩa hàm đọc dữ liệu

def readData(bounding\_boxes):

    try:

        #print(bounding\_boxes)

        dark\_size = [bounding\_boxes[i+1][0] - bounding\_boxes[i][0] - bounding\_boxes[i][2] for i in range(len(bounding\_boxes)-1)]

        long\_size = max(dark\_size)

        index =[]

        sum\_header\_size = 0

        for i,value in enumerate(dark\_size):

            if value >= 3/4 \*long\_size:

                index.append(bounding\_boxes[i][0]+bounding\_boxes[i][2])

        for i in range(len(index)-1):

            sum\_header\_size += index[i+1]-index[i]

        if len(index) >=3:

            bit\_size = round(sum\_header\_size/ (len(index)-1) / 11)

        else:

            bit\_size = round(sum\_header\_size / 11)

        bit = []

        bit\_list = []

        for i in range(1,len(bounding\_boxes)):

            if 3 / 4 \* long\_size < dark\_size[i-1] <= long\_size:

                print(bit)

                if len(bit) == length\_data+2:

                    bit\_list.append(bit)

                bit = []

            else:

                if bit:

                    for \_ in range(math.ceil((dark\_size[i-1] / bit\_size))):

                        bit.append(0)

            for \_ in range(bounding\_boxes[i][2]//bit\_size):

                bit.append(1)

        if len(bit\_list) > 0:

            bit = bit\_list[len(bit\_list)//2]

        else:

            bit = []

        return bit

    except:

        return []

# định nghĩa hàm giải mã dữ liệu

def decoder(data\_decoder):

    # tiền xử lí

    data = []

    for bit in reversed(data\_decoder[1:-1]):

        data.append(bit)

    # tiến hành giải mã

    sk = 0

    r = number\_one

    for i in range(len(data) - 1, -1, -1):

        if data[i] == 1:

            if i+1 > r:

                sk = sk + math.comb(i, r)

            r -=1

    data\_4bit = format(sk, '04b')

    return data\_4bit

# định nghĩa hàm trích xuất RoI

def grid\_division(image, nLED,threshold\_read,width,camera):

    \_, threshold\_image = cv2.threshold(image, threshold\_read, 255, cv2.THRESH\_BINARY  )

    contours, \_ = cv2.findContours(threshold\_image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

    # Tính kích thước trung bình của các hình chữ nhật

    average\_size = np.mean([cv2.contourArea(contour) for contour in contours])

    bounding\_boxes = []

    # Vẽ hình chữ nhật bao quanh mỗi vân sáng trên ảnh

    for contour in contours:

        # Tính kích thước của hình chữ nhật

        area = cv2.contourArea(contour)

        if area > average\_size / 10:

            # Chỉ vẽ hình chữ nhật nếu kích thước lớn hơn 1/10 kích thước trung bình

            x, y, w, h = cv2.boundingRect(contour)

            # Lưu trữ bộ số trong danh sách

            bounding\_boxes.append((x, y, w, h))

    center\_array =[]

    led\_area = 0

    for box in bounding\_boxes:

        x,y,w,h = box

        if h > led\_area:

            led\_area = h

        center = (x + w // 2, y + h // 2)

        center\_array.append(center)

    # Áp dụng thuật toán DBSCAN

    dbscan = DBSCAN(eps=round(width\*0.1/2), min\_samples=2)

    labels = dbscan.fit\_predict(center\_array)

    n = max(labels)

    data=[]

    rois=[]

    local=[]

    m = width//led\_area

    if n == (nLED-1):

        list\_of\_contours = [[] for \_ in range(nLED)]

        data=[[] for \_ in range(nLED)]

        for index, value in enumerate(labels):

            if value >=0:

                list\_of\_contours[value].append(bounding\_boxes[index])

        array\_local =[2\*width, width]

        for  index,contour in enumerate(list\_of\_contours):

            contour.sort(key=lambda box: box[0])

            data[index] = readData(contour)

            rois.append([contour[0][0],min(contour, key=lambda arr: arr[1])[1], contour[-1][0]-contour[0][0]+ contour[-1][2],max(contour, key=lambda arr: arr[3])[3]])

            if rois[index][0] < array\_local[0]:

                array\_local[0] = rois[index][0]

            if rois[index][1] < array\_local[1]:

                array\_local[1] = rois[index][1]

        for roi in rois:

            local.append((roi[1]-array\_local[1])//(1.5\*led\_area) \* m + (roi[0]-array\_local[0])//(1.5\*led\_area) )

        if (camera == 1):

            for i in range(n+1):

                local[i] = (local[i]//m+1)\*m - local[i]%m - 1

        # Sắp xếp theo chỉ số đầu tiên của mỗi mảng trong list\_of\_arrays1

        sorted\_lists = sorted(zip(local,rois, data), key=lambda x: x[0])

        # Tách lại danh sách mảng 1 và mảng 2

        \_,rois, data = zip(\*sorted\_lists)

    return data,rois